PATENT.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of: Neil W. Black, et al.	Atty. Dkt. No.: 003797.86776
Appln. No.: 09/592,660	Group Art Unit: 3628
Filed: June 13, 2000	Examiner: F. Poinvil
For: SYSTEM AND METHOD FOR CREATING MODEL INVESTMENT PORTFOLIOS	

DECLARATION UNDER 37 C.F.R. § 1.131

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

I, Neil W. Black, do hereby declare as follows:

1. I am named as an inventor in U.S. Patent Appln. No. 09/592,660, filed June 13, 2000, along with Peter Hansen and Jon D. Markman. I am an employee of Microsoft Corporation of Redmond, Washington, and I am over the age of eighteen years.

2. From our company's records, I located and reviewed computer code entitled "setupwiz." A true and correct hard copy of that computer code is attached as Exhibit A. The computer system from which this specific copy of the "setupwiz" computer code was obtained indicated that the code had last been modified (*i.e.*, its "last modified date") at some time prior to August 11, 1999.

3. Accordingly, to the best of my knowledge and belief, the "setupwiz" code existed in the form shown in Exhibit A prior to August 11, 1999.

4. From our company's records, I also located and reviewed computer code entitled "setupwiz.h." A true and correct hard copy of that computer code is attached as Exhibit B. The computer system from which this specific copy of the "setupwiz.h" computer code was obtained indicated that the code had last been modified (*i.e.*, its "last modified date") at some time prior to August 11, 1999.

5. Accordingly, to the best of my knowledge and belief, the "setupwiz.h" code existed in the form shown in Exhibit B prior to August 11, 1999.

6. The "setupwiz" and "setupwiz.h" codes, in the form shown in Exhibits A and B, respectively, were used as part of a beta test of the invention that began prior to August 11, 1999. Using these codes, the invention performed in its intended manner, as described in our patent application and its claims, and thus was actually reduced to practice prior to August 11, 1999.

DECLARATION IN LIEU OF OATH

7. I further declare that all information stated herein based upon my own knowledge is true and that all information stated herein based on information and belief is believed to be true, and further that the statements made in this Declaration were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of this application or any patent issuing based on this application.

Date: 8/3/04 By: Neil W. Black
Neil W. Black, Inventor

EXHIBIT A

Copy of "setupwiz" Computer Code

```
#include "global.h"
#include "dbwin.h"
#include "resrc1.h"
#include "format.h"
#include "buydlg.h"
#include "import.h"
#include "quicken.h"
#include "ofxsess.h"
#include "ofx.h"
#include "ofxutil.h"
#include "security.h"
#include "acctdlg.h"
#include "tickmisc.h"
#include "setupwiz.h"
#include "internet.h"
#include "monthdlg.h"
#include "findsym.h"
#include "roaming.h"
#include "ofxprofile.h"
#include "ofxsignup.h"
#include "propsht.h"
SZTHISFILE

NAW_TIP_STUCT rgTips[] = {
        {IDS_NAW_TIP_ACCT,              IDS_PREF_NO_NAW_TIP_ACCT,
IDB_RECORD_ACCT_TIP},
        {IDS_NAW_TIP_INV,               IDS_PREF_NO_NAW_TIP_INV,
IDB_RECORD_BUY_TIP},
        {IDS_NAW_TIP_ROAM,              IDS_PREF_NO_NAW_TIP_ROAM,
IDB_RECORD_ROAM_TIP},
        };

//----------------------------------------------------------------------------
-------
// Setup Wizard Dialog Utilities
//----------------------------------------------------------------------------
-------
void SetFirstOFXPropSheet(HWND hwndDlg, SETUP_STRUCT *pSetup)
{
        TCHAR   szPassword[MAX_PASSWORD];

        pSetup->pOCX->GetMasterPassword(szPassword, MAX_PASSWORD);

        if (FileHasEncryptedOFXData(pSetup->pOCX)
                || (pSetup->pOCX->GetSecurityLevel() == SECURITY_HIGH) ||
lstrlen(szPassword))
        {
                assert(pSetup->pOFX);
                pSetup->pOFX->SetPasswordDlgParent(hwndDlg);

                if (pSetup->pOFX->InitOFX(FALSE))
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_OFX_1);
                else
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
        }
        else
                SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_MASTER_PASSWORD);
}

//----------------------------------------------------------------------------
-------
// Checks to see if there is data to convert when user clicks next
```

```
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizFTUEDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_NEXT);
                        break;

                case PSN_WIZNEXT:
                        // Setup the sample portfolios.
                        return TRUE;

                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)((INVPROPSHEETPAGE *)lParam)->lParam;
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
                return TRUE;
        }

        return(FALSE);
}


//-------------------------------------------------------------------------------
-------
// Dialog proc for the setup wizard New Account Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizNewDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam, LPARAM
lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
        int watchFlag = 0;

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_COMMAND:
                switch(HIWORD(wParam))
                {
                case BN_DBLCLK:
                        PropSheet_PressButton(GetParent(hwndDlg), PSBTN_NEXT);
                        break;
```

```
                }

                switch(LOWORD(wParam))
                {
                case IDC_ACCOUNT_REGULAR:
                        SetDlgItemText(hwndDlg, IDC_ACCOUNT_DESCRIPTION,
GetRscString(IDS_ACCT_DESC_REGULAR));
                        break;

                case IDC_ACCOUNT_WATCH:
                        SetDlgItemText(hwndDlg, IDC_ACCOUNT_DESCRIPTION,
GetRscString(IDS_ACCT_DESC_WATCH));
                        break;

                case IDC_ACCOUNT_IMPORT_EXISTING:
                        SetDlgItemText(hwndDlg, IDC_ACCOUNT_DESCRIPTION,
GetRscString(IDS_ACCT_DESC_IMPORT));
                        break;
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_NEXT);
                        break;

                case PSN_WIZNEXT:
                        SW_ACCT_TYPE            swatSelect;

                        swatSelect = SWAT_NONE;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_REGULAR))
                                swatSelect = SWAT_REGULAR;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_WATCH))
                                swatSelect = SWAT_WATCH;

                        if (IsDlgButtonChecked(hwndDlg,
IDC_ACCOUNT_IMPORT_EXISTING))
                                swatSelect = SWAT_IMPORT;

                        switch (swatSelect)
                        {
                        case SWAT_REGULAR:
                                if (pSetup->pAds == NULL)
                                {
                                        pSetup->pAds = new ACCTDLG_STRUCT;
                                        assert(pSetup->pAds);
                                        my_memset(pSetup->pAds, 0,
sizeof(ACCTDLG_STRUCT));
                                }

                                pSetup->pAds->fWatch = FALSE;

                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_REGULAR);
                                break;

                        case SWAT_WATCH:
```

```
                                        setupwiz
                                if (pSetup->pAds == NULL)
                                {
                                        pSetup->pAds = new ACCTDLG_STRUCT;
                                        assert(pSetup->pAds);
                                        my_memset(pSetup->pAds, 0,
sizeof(ACCTDLG_STRUCT));
                                }

                                pSetup->pAds->fWatch = TRUE;

                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_WATCH);
                                break;

                        case SWAT_IMPORT:
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
                                break;

                        default:
                                assert(FALSE);
                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                                break;
                        }

                        pSetup->swatSelect = swatSelect;

                        return TRUE;
                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                CheckDlgButton(hwndDlg, IDC_ACCOUNT_REGULAR, BST_CHECKED);

                SetDlgItemText(hwndDlg, IDC_ACCOUNT_DESCRIPTION,
GetRscString(IDS_ACCT_DESC_REGULAR));

                if (GET_ROAMING(pSetup->pOCX)->IsSavingData())
                        EnableControl(hwndDlg, IDC_ACCOUNT_IMPORT_EXISTING, FALSE);

                return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------
-------
// Dialog proc for the setup wizard Import Existing Account Property Sheet.
//-------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizImportExistingDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
        RECT                    rcTop, rcBottom, rcSquare, rcQuickenDC, rcOFX;
        int                     iOffset = 0;

        switch(uMsg)
        {
```

```
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_COMMAND:
                switch(HIWORD(wParam))
                {
                case BN_DBLCLK:
                        PropSheet_PressButton(GetParent(hwndDlg), PSBTN_NEXT);
                        break;
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
                        break;

                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_NEW);
                        return TRUE;

                case PSN_WIZNEXT:
                        SW_ACCT_TYPE            swatSelect;

                        swatSelect = SWAT_IMPORT;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_MONEY))
                                swatSelect = SWAT_MONEY;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_QUICKEN))
                                swatSelect = SWAT_QUICKEN;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_YAHOO))
                                swatSelect = SWAT_YAHOO;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM))
                                swatSelect = SWAT_QUICKEN_COM;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_OFX))
                                swatSelect = SWAT_OFX;

                        switch (swatSelect)
                        {
                        case SWAT_MONEY:
                                if (pSetup->pMoney == NULL){
                                        pSetup->pMoney = new CMoney(pSetup->pOCX,
GET_DATA(pSetup->pOCX));
                                        assert(pSetup->pMoney);
                                        pSetup->pMoney->AddRef();

pSetup->pMoney->SetHWndParent(GetParent(hwndDlg));
                                }
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_MONEY_1);
                                break;
```

```
                        case SWAT_QUICKEN:
                                if (pSetup->pQuicken == NULL){
                                        pSetup->pQuicken = new
CQuicken(pSetup->pOCX, GET_DATA(pSetup->pOCX));
                                        assert(pSetup->pQuicken);
                                        pSetup->pQuicken->AddRef();

pSetup->pQuicken->SetHWndParent(GetParent(hwndDlg));
                                }
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_QUICKEN);
                                break;

                        case SWAT_YAHOO:
                        case SWAT_QUICKEN_COM:
                                if (pSetup->pWeb == NULL){
                                                pSetup->pWeb = new
CWeb(pSetup->pOCX, GET_DATA(pSetup->pOCX));
                                                assert(pSetup->pWeb);
                                                pSetup->pWeb->AddRef();

pSetup->pWeb->SetHWndParent(GetParent(hwndDlg));
                                }

                                if (swatSelect == SWAT_YAHOO)
                                        pSetup->pWeb->setOC(YAHOO);
                                else if (swatSelect == SWAT_QUICKEN_COM)
                                        pSetup->pWeb->setOC(QUICKENDC);
                                else
                                        assert(FALSE);

                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_WEB);
                                break;

                        case SWAT_OFX:
                                if (pSetup->pOFX == NULL)
                                {
                                        pSetup->pOFX = new COFX(pSetup->pOCX,
GET_DATA(pSetup->pOCX));
                                        assert(pSetup->pOFX);
                                        pSetup->pOFX->AddRef();

pSetup->pOFX->SetHWndParent(GetParent(hwndDlg));
                                }

                                SetFirstOFXPropSheet(hwndDlg, pSetup);
                                break;

                        default:
                                assert(FALSE);
                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                                break;
                        }

                        pSetup->swatSelect = swatSelect;
                        return TRUE;
                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
```

```
                //should we resize?
                if (pSetup->pOCX->IsYahooEnabled() == FALSE ||
                        pSetup->pOCX->IsQuickenDotComEnabled() == FALSE) {
                        if ((pSetup->pOCX->IsYahooEnabled() == FALSE) &&
                                (pSetup->pOCX->IsQuickenDotComEnabled() == FALSE)) {
                                HideControl(hwndDlg, IDC_ACCOUNT_YAHOO);
                                HideControl(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM);

                                GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM,
&rcBottom);
                                ScreenToClient(hwndDlg, (POINT *)&rcBottom);

                                GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKEN,
&rcTop);
                                ScreenToClient(hwndDlg, (POINT *)&rcTop);
                        } else if (pSetup->pOCX->IsYahooEnabled() == FALSE) {
                                HideControl(hwndDlg, IDC_ACCOUNT_YAHOO);

                                GetControlRect(hwndDlg, IDC_ACCOUNT_YAHOO,
&rcBottom);
                                ScreenToClient(hwndDlg, (POINT *)&rcBottom);

                                GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKEN,
&rcTop);
                                ScreenToClient(hwndDlg, (POINT *)&rcTop);

                                GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM,
&rcQuickenDC);
                                ScreenToClient(hwndDlg, (POINT *)&rcQuickenDC);
                        } else if (pSetup->pOCX->IsQuickenDotComEnabled() == FALSE)
{
                                HideControl(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM);

                                GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM,
&rcBottom);
                                ScreenToClient(hwndDlg, (POINT *)&rcBottom);

                                GetControlRect(hwndDlg, IDC_ACCOUNT_YAHOO, &rcTop);
                                ScreenToClient(hwndDlg, (POINT *)&rcTop);
                        }
                        // Find out how far to move the items.
                        iOffset = rcBottom.bottom - rcTop.bottom;

                        GetControlRect(hwndDlg, IDC_IMPORT_SQUARE, &rcSquare);

                        if (pSetup->pOCX->IsQuickenDotComEnabled() != FALSE) {
                                // Move QuickenDotCom up if needed
                                SetWindowPos(GetDlgItem(hwndDlg,
IDC_ACCOUNT_QUICKENDOTCOM), NULL,
                                                rcQuickenDC.left, rcQuickenDC.top -
iOffset, 0, 0,
                                                SWP_NOZORDER | SWP_NOSIZE |
SWP_NOACTIVATE);
                        }

                        // Move OFX up if needed
                        GetControlRect(hwndDlg, IDC_ACCOUNT_OFX, &rcOFX);
                        ScreenToClient(hwndDlg, (POINT *)&rcOFX);
                        SetWindowPos(GetDlgItem(hwndDlg, IDC_ACCOUNT_OFX), NULL,
                                        rcOFX.left, rcOFX.top - iOffset, 0, 0,
                                        SWP_NOZORDER | SWP_NOSIZE |
SWP_NOACTIVATE);
```

```
                        // Resize the frame
                        SetWindowPos(GetDlgItem(hwndDlg, IDC_IMPORT_SQUARE), NULL,
0, 0,
                                rcSquare.right - rcSquare.left, rcSquare.bottom -
rcSquare.top - iOffset,
                                SWP_NOZORDER | SWP_NOMOVE | SWP_NOACTIVATE);

                }
                CheckDlgButton(hwndDlg, IDC_ACCOUNT_MONEY, BST_CHECKED);

                return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Regular Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK DoesEditBoxHaveText(HWND hwndDlg, long idcItem)
{
        TCHAR   szText[MAX_STRING];
        BOOL    fResult = FALSE;

        szText[0] = 0;
        if (GetDlgItemText(hwndDlg, idcItem, szText, MAX_STRING))
                alltrim(szText);

        if (strlen(szText) > 0)
                fResult = TRUE;

        return(fResult);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Regular Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizRegularDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_USER_SETFOCUS:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                if (pSetup->pAds->fErrorDirty)
                        DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
                pSetup->pAds->fErrorDirty = FALSE;
                break;

        case WM_COMMAND:
                switch(HIWORD(wParam))
                {
                case EN_CHANGE:
                        if (DoesEditBoxHaveText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME))
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
                        else
```

```
                        PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
                        break;
                }

                switch(LOWORD(wParam))
                {
                case IDC_NEWACCT_CASH:
                        goto lbEnableCashBalance;
                        break;
                }
                break;

        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        if (DoesEditBoxHaveText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME))
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
                        else
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);

lbEnableCashBalance:
                        if (IsDlgButtonChecked(hwndDlg, IDC_NEWACCT_CASH))
                        {
                                EnableControl(hwndDlg, IDC_NEWACCT_CASHBAL, TRUE);
                                EnableHotKey(hwndDlg, IDC_NEWACCT_CASHBAL_TEXT, 'I',
TRUE);

                        }
                        else
                        {
                                EnableControl(hwndDlg, IDC_NEWACCT_CASHBAL, FALSE);
                                EnableHotKey(hwndDlg, IDC_NEWACCT_CASHBAL_TEXT, 'I',
FALSE);

                        }
                        break;

                case PSN_WIZFINISH:
                        break;

                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_NEW);
                        return TRUE;

                case PSN_WIZNEXT:
                        int                     resIdError = IDS_NEED_ACCOUNT_NAME;
                        int                     iBadControl =
IDC_NEWACCT_ACCOUNT_NAME;
                        BOOL            fSuccess = FALSE;
                        double          dbl = 0;

                        // Validate the Account Name.
```

```
                        if (GetDlgItemText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME,
pSetup->pAds->szAccountName, MAX_ACCOUNT_NAME))
                        {
                                // Make sure we have a name
                                alltrim(pSetup->pAds->szAccountName);
                                fSuccess = lstrlen(pSetup->pAds->szAccountName) > 0;
                        }

                        // Check if account already exists
                        if (fSuccess)
                        {
                                resIdError = IDS_ACCOUNT_NAME_IN_USE;
                                fSuccess =
GET_DATA(pSetup->pOCX)->FindAccountName(pSetup->pAds->szAccountName) == NULL;
                        }

                        // Validatation Cash balance for basic format -- numerals,
commas, etc.
                        if (fSuccess)
                        {
                                iBadControl = IDC_NEWACCT_CASHBAL;

                                fSuccess = DlgValidateQuantity(hwndDlg,
IDC_NEWACCT_CASHBAL, &resIdError);

                                // get the purchase price
                                if (fSuccess)
                                        fSuccess = DlgGetDouble(hwndDlg,
IDC_NEWACCT_CASHBAL, BDGD_MAX_LONG, &dbl, &resIdError);
                        }

                        // If we're still ok, go to the finish screen, otherwise
error
                        if (fSuccess)
                        {
                                pSetup->pAds->dblCash = dbl;
                                pSetup->pAds->fCash      =
IsDlgButtonChecked(hwndDlg, IDC_NEWACCT_CASH);
                                PropSheet_PressButton(GetParent(hwndDlg),
PSBTN_FINISH);
                        }
                        else
                        {
                                pSetup->pAds->fErrorDirty = TRUE;
                                my_beep();
                                PostMessage(hwndDlg, WM_USER_SETFOCUS,
(WPARAM)iBadControl, resIdError);
                        }

                        // Don't let the next button take us anywhere...
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                        return(TRUE);
                }
                break;


        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                SubClassNumberField(hwndDlg, IDC_NEWACCT_CASHBAL, FALSE);

                // Set label on Finish button to Next
```

```
                return(TRUE);
        }

        return(FALSE);
}


//--------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Regular Property Sheet.
//--------------------------------------------------------------------------------
-------
void CheckWatchAccountButtonStatus(HWND hwndDlg)
{
        if ((DoesEditBoxHaveText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME)) &&
                (DoesEditBoxHaveText(hwndDlg, IDC_WATCH_SYMBOLS)))
                PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_FINISH);
        else
                PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_DISABLEDFINISH);
}

//--------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Regular Property Sheet.
//--------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizWatchDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_USER_SETFOCUS:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                if (pSetup->pAds->fErrorDirty)
                        DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
                pSetup->pAds->fErrorDirty = FALSE;
                break;

        case WM_COMMAND:
                switch(HIWORD(wParam))
                {
                case EN_CHANGE:
                        CheckWatchAccountButtonStatus(hwndDlg);
                        break;
                }

                switch(LOWORD(wParam))
                {
                case IDCANCEL:
                        // Wierd: v7 bug 2186: Multiline edit control does not pass
                        //  the cancel method up to the parent, so we have to catch
it here
                        //  and force it ourselves.
                        PropSheet_PressButton(GetParent(hwndDlg), PSBTN_CANCEL);
                        break;

                case IDC_WATCH_FIND_SYMBOL:
                        pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
```

```
SETUPPROP_PROP_NAME);

                        // Call up the find symbol dialog.  It returns TRUE if the
user pressed
                        // OK to exit it. szTicker is the ticker selected by the
user in the Find
                        // Symbol dialog.
                        {
                                CLISTATUS status;
                                OCX(pSetup->pOCX)->InitWosaStatusStruct(&status,
qrsPortFindSymbol);

                                TCHAR   szSymbols[MAX_STRING * 2] = {0};
                                TCHAR   szTicker[MAX_STRING] = {0};
                                int             cch;

                                if (FindSymbolDlg(hwndDlg, szTicker,
OCX(pSetup->pOCX)->GetDefaultTickerCurid(),
                                        ((CPortfolioControl
*)pSetup->pOCX)->m_spClientSite, &status, TRUE,
                                        FALSE, INV_STOCK)
                                    && lstrlen(szTicker))
                                {
                                        SendDlgItemMessage(hwndDlg,
IDC_WATCH_SYMBOLS, WM_GETTEXT, MAX_STRING, (LPARAM)szSymbols);
                                        alltrim(szSymbols);

                                        if (lstrlen(szSymbols) > 0)
                                                StrCatMaxLen(szSymbols, ", ",
sizeof(szSymbols));

                                        StrCatMaxLen(szSymbols, szTicker,
sizeof(szSymbols));

                                        // Chop off the szSymbols to fit into 256
characters..
                                        if (lstrlen(szSymbols) > 256)
                                        {
                                                cch = 255;

                                                for (cch = 255; (cch > 0) &&
(szSymbols[cch] != ','); cch--)
                                                        szSymbols[cch] = '\0';
                                        }

                                        SendDlgItemMessage(hwndDlg,
IDC_WATCH_SYMBOLS, WM_SETTEXT, MAX_STRING, (LPARAM)szSymbols);
                                        CheckWatchAccountButtonStatus(hwndDlg);
                                }
                        }
                        break;

                case IDC_WATCH_ADVANCED:
                        if (IsDlgButtonChecked(hwndDlg, IDC_WATCH_ADVANCED))
                        {
                                ShowWindow(GetDlgItem(hwndDlg, IDC_MODEL_OPTIONS),
SW_SHOW);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_DOLLARS),
SW_SHOW);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_SHARES),
SW_SHOW);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_TOTAL),
SW_SHOW);
```

```
                                        setupwiz
                                ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_TOTAL_EDIT), SW_SHOW);
                                ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_DATE_CAPTION), SW_SHOW);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_DATE),
SW_SHOW);
                                if (!g_fOldComCtl)
                                        ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_CALICON), SW_SHOW);

                                ShowWindow(GetDlgItem(hwndDlg,
IDC_ADVANCED_DESCRIPTION), SW_HIDE);
                        }
                        else
                        {
                                ShowWindow(GetDlgItem(hwndDlg, IDC_MODEL_OPTIONS),
SW_HIDE);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_DOLLARS),
SW_HIDE);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_SHARES),
SW_HIDE);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_TOTAL),
SW_HIDE);
                                ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_TOTAL_EDIT), SW_HIDE);
                                ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_DATE_CAPTION), SW_HIDE);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_DATE),
SW_HIDE);
                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_CALICON),
SW_HIDE);

                                ShowWindow(GetDlgItem(hwndDlg,
IDC_ADVANCED_DESCRIPTION), SW_SHOW);
                        }
                        break;

                case IDC_WATCH_DOLLARS:
                case IDC_WATCH_SHARES:
                case IDC_WATCH_TOTAL:
                        if (IsDlgButtonChecked(hwndDlg, IDC_WATCH_TOTAL))
                        {
                                EnableControl(hwndDlg, IDC_WATCH_TOTAL_EDIT, TRUE);
                        }
                        else
                        {
                                EnableControl(hwndDlg, IDC_WATCH_TOTAL_EDIT, FALSE);
                        }
                        break;

                case IDC_WATCH_CALICON:
                        DoMonthDlg(hwndDlg, IDC_WATCH_DATE, IDC_WATCH_CALICON);
                        break;
                }
                break;

        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
```

```
        assert(pSetup);
        switch (((NMHDR *)lParam)->code)
        {
        case PSN_SETACTIVE:
                CheckWatchAccountButtonStatus(hwndDlg);
                break;

        case PSN_WIZBACK:
                SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_NEW);
                return TRUE;

        case PSN_WIZFINISH:
                int                     resIdError = IDS_NEED_ACCOUNT_NAME;
                int                     iBadControl =
IDC_NEWACCT_ACCOUNT_NAME;
                BOOL            fSuccess = FALSE;
                double          dbl = 0;

                // Validate the Account Name.
                if (GetDlgItemText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME,
pSetup->pAds->szAccountName, MAX_ACCOUNT_NAME))
                {
                        // Make sure we have a name
                        alltrim(pSetup->pAds->szAccountName);
                        fSuccess = lstrlen(pSetup->pAds->szAccountName) > 0;
                }

                // Check if account already exists
                if (fSuccess)
                {
                        resIdError = IDS_ACCOUNT_NAME_IN_USE;
                        fSuccess =
GET_DATA(pSetup->pOCX)->FindAccountName(pSetup->pAds->szAccountName) == NULL;
                }

                if (fSuccess)
                {
                        resIdError = IDS_ACCOUNT_NEED_SYMBOLS;
                        iBadControl = IDC_WATCH_SYMBOLS;

                        // Get list of symbols
                        if (GetDlgItemText(hwndDlg, IDC_WATCH_SYMBOLS,
pSetup->pAds->szSymbols, MAX_STRING))
                        {
                                // Make sure we have a name
                                alltrim(pSetup->pAds->szSymbols);
                                fSuccess = lstrlen(pSetup->pAds->szSymbols)
> 0;
                        }
                }

                // Get advanced options if visible.
                if (fSuccess)
                {
                        if (IsDlgButtonChecked(hwndDlg, IDC_WATCH_ADVANCED))
                        {
                                pSetup->pAds->fModel = TRUE;

                                if (IsDlgButtonChecked(hwndDlg,
IDC_WATCH_DOLLARS))
                                        pSetup->pAds->mo = MO_DOLLARS;

                                if (IsDlgButtonChecked(hwndDlg,
```

```
                                        setupwiz
IDC_WATCH_SHARES))
                                                pSetup->pAds->mo = MO_SHARES;

                                        if (IsDlgButtonChecked(hwndDlg,
IDC_WATCH_TOTAL))
                                        {
                                                TCHAR   szTotal[MAX_STRING];

                                                iBadControl = IDC_WATCH_TOTAL_EDIT;
                                                resIdError = IDS_NEED_POSITIVE;
                                                fSuccess = FALSE;

                                                pSetup->pAds->mo = MO_TOTAL;
                                                if (GetDlgItemText(hwndDlg,
IDC_WATCH_TOTAL_EDIT, szTotal, MAX_STRING))
                                                {
                                                        // Make sure we have a name
                                                        alltrim(szTotal);
                                                        pSetup->pAds->dblMOTotal =
my_atol(szTotal);

                                                        fSuccess =
pSetup->pAds->dblMOTotal > 0;
                                                }
                                        }

                                        if (fSuccess)
                                        {
                                                CDate                   date, today;
                                                SYSTEMTIME              st;

                                                GetLocalTime(&st);
                                                today.SetDate(&st);

                                                resIdError = IDS_NEED_MODEL_DATE;
                                                iBadControl = IDC_WATCH_DATE;

                                                GetDlgItemText(hwndDlg,
IDC_WATCH_DATE, pSetup->pAds->szMODate, MAX_STRING);

                                                if (!SetDateToString(&date,
pSetup->pAds->szMODate, NULL))
                                                        fSuccess = FALSE;
                                                else if (date > today)
                                                        fSuccess = FALSE;
                                        }
                                }
                        }

                        // If we're not ok, error
                        if (!fSuccess)
                        {
                                pSetup->pAds->fErrorDirty = TRUE;
                                my_beep();
                                PostMessage(hwndDlg, WM_USER_SETFOCUS,
(WPARAM)iBadControl, resIdError);
                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                        }
                        return(TRUE);
                }
                break;

        case WM_INITDIALOG:
```

```
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                // Set the sample tickers
                // SetDlgItemText(hwndDlg, IDC_WATCH_SYMBOLS, "MSFT, INTC, DELL");

                // Set the default option
                CheckDlgButton(hwndDlg, IDC_WATCH_SHARES, BST_CHECKED);

                // Subclass number fields
                SubClassNumberField(hwndDlg, IDC_WATCH_TOTAL_EDIT, FALSE);

                // Set the total portolio value to $10 000
                SetDlgItemText(hwndDlg, IDC_WATCH_TOTAL_EDIT, "10000");

                // Set up the calendar bitmap button
                SetCalendarButton(hwndDlg, IDC_WATCH_CALICON, IDB_CALICON);

                // Subclass the date field to enable +/- keys
                SubClassDateEditField(hwndDlg, IDC_WATCH_DATE);

                // Set the date initially to today.
                {
                        SYSTEMTIME      st;
                        TCHAR           szTemp[MAX_STRING];
                        GetLocalTime(&st);
                        CDate           today;

                        today.SetDate(&st);

                        SetDlgItemText(hwndDlg, IDC_WATCH_DATE,
today.Format(szTemp));
                }

                // Check for enabling/disabling of advanced checkbox:
IDC_WATCH_ADVANCED

                return(TRUE);

        case WM_DESTROY:
                FreeCalendarBitmap(hwndDlg);
                return(0);

        }

        return(FALSE);
}

//-----------------------------------------------------------------------------
-------
// Dialog proc for the Acocunt Wizard Finish Property Sheet. (Shared by all account
types)
//-----------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
```

```
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_FINISH);
                        break;

                case PSN_WIZBACK:
                        // We don't know where we are going - the calling dialog
proc should handle this message.
                        assert(FALSE);
                        break;

                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
                return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Web Password Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizWebDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam, LPARAM
lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
        TCHAR                   tmpID[MAX_STRING];
        TCHAR                   tmpPassword[MAX_STRING];

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_USER_WEBTXT_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup->pWeb);

                if ((DoesEditBoxHaveText(hwndDlg, IDC_ACCOUNT_WEB_USERID)) &&
                        (DoesEditBoxHaveText(hwndDlg, IDC_ACCOUNT_WEB_PASSWORD)))
                {
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
                }
                else
                {
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
```

```
                }
                break;

        case WM_COMMAND:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch(HIWORD(wParam))
                {
                case EN_CHANGE:
                        if ((pSetup->pWeb->isDownloaded() != NOTHING) &&
                                (DoesEditBoxHaveText(hwndDlg,
IDC_ACCOUNT_WEB_USERID)) &&
                                (DoesEditBoxHaveText(hwndDlg,
IDC_ACCOUNT_WEB_PASSWORD)))
                        {
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
                        }
                        else
                        {
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
                        }
                        break;
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        assert(pSetup->pWeb);

                        pSetup->pWeb->SetWebPrompts(hwndDlg);

                        if ((pSetup->pWeb->EnsureWebTxt(NULL, hwndDlg))
                                || (!DoesEditBoxHaveText(hwndDlg,
IDC_ACCOUNT_WEB_USERID))
                                || (!DoesEditBoxHaveText(hwndDlg,
IDC_ACCOUNT_WEB_PASSWORD)))
                        {
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
                        }
                        else
                        {
                                PropSheet_SetWizButtons(GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
                        }
                        break;

                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
                        return TRUE;

                case PSN_WIZNEXT:
                        //Get user name and password
                        GetDlgItemText(hwndDlg, IDC_ACCOUNT_WEB_USERID, tmpID,
MAX_STRING);
                        GetDlgItemText(hwndDlg, IDC_ACCOUNT_WEB_PASSWORD,
```

```
tmpPassword, MAX_STRING);
                        alltrim(tmpID);
                        alltrim(tmpPassword);
                        if (my_strcmp(tmpID, pSetup->pWeb->checkID())
                                || my_strcmp(tmpPassword,
pSetup->pWeb->checkPassword())) {
                                pSetup->pWeb->ClearAccounts();
                        }
                        pSetup->pWeb->setUserID(tmpID);
                        pSetup->pWeb->setPassword(tmpPassword);
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_WEB_LIST);
                        return(TRUE);

                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                assert(pSetup);

                return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Money 2 Property Sheet.
//-------------------------------------------------------------------------------
-------
void CheckImportWizButtonStatus(SETUP_STRUCT *pSetup, HWND hwndDlg)
{
        // Check the selection
        if (SendDlgItemMessage(hwndDlg, IDC_ACCOUNT_LIST, LB_GETSELCOUNT, 0, 0) > 0)
        {
                if (pSetup->fNext == FALSE)
                {
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
                        pSetup->fNext = TRUE;
                }
        }
        else
        {
                if (pSetup->fNext == TRUE)
                {
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
                        pSetup->fNext = FALSE;
                }
        }
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Web Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizWebListDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
```

```
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_USER_PORT_LIST_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup->pWeb);

                // Populate the list with the accounts.
                pSetup->pWeb->FillAccountList(hwndDlg, NULL);

                // Update the navigation elements.
                CheckImportWizButtonStatus(pSetup, hwndDlg);

                break;

        case WM_USER_BAD_LOGIN:
                PropSheet_PressButton(GetParent(hwndDlg), PSBTN_BACK);
                        break;

        case WM_COMMAND:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                switch(LOWORD(wParam))
                {
                case IDC_ACCOUNT_LIST:
                        if (HIWORD(wParam) == LBN_SELCHANGE)
                        {
                                CheckImportWizButtonStatus(pSetup, hwndDlg);
                        }
                        break;
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        assert(pSetup->pWeb);
                        pSetup->fNext = FALSE;

                        switch (pSetup->pWeb->checkOC())
                        {
                        case YAHOO:
                                SetDlgItemText(hwndDlg, IDC_DOWNLOAD_CAPTION,
GetRscString(IDS_WEB_YAHOO_PORTFOLIOS));
                                break;

                        case QUICKENDC:
                                SetDlgItemText(hwndDlg, IDC_DOWNLOAD_CAPTION,
GetRscString(IDS_WEB_QUICKEN_PORTFOLIOS));
                                break;

                        default:
                                Assert(FALSE);
                        }
```

```
                        if(pSetup->pWeb->EnsurePortfolioList(NULL, hwndDlg))
                        {
                                SendDlgItemMessage(hwndDlg, IDC_ACCOUNT_LIST,
LB_RESETCONTENT, 0, 0);
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
                        }
                        else
                        {
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
                                CheckImportWizButtonStatus(pSetup, hwndDlg);
                        }

                        break;

                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_WEB);
                        return TRUE;

                case PSN_WIZNEXT:
                        // BUG BUG BUG! Should be ensuring an item is selected
before getting here.
                        if (pSetup->pWeb->GetAccountsSelected(hwndDlg))
                        {
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_WEB_FINISH);
                        }
                        else
                        {
                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                        }
                        return(TRUE);

                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                assert(pSetup);

                return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Yahoo Finish Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizWebFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
```

```
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        assert(pSetup->pWeb);

                        if (pSetup->pWeb->checkOC() == QUICKENDC) {
                                SetDlgItemText(hwndDlg, IDC_DOWNLOAD_CAPTION,
"Downloading Quicken.com information...");
                        }
                        break;
                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_WEB_LIST);
                        return TRUE;
                }
                break;
        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                assert(pSetup);

                return(TRUE);
        }
        return(AccountWizFinishDialogProc(hwndDlg, uMsg, wParam, lParam));
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Money 1 Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizMoney1DialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_COMMAND:
                switch(HIWORD(wParam))
                {
                case BN_DBLCLK:
                        PropSheet_PressButton(GetParent(hwndDlg), PSBTN_NEXT);
                        break;
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
                        break;
```

```
                case PSN_WIZNEXT:
                        SW_ACCT_TYPE            swatSelect;

                        swatSelect = SWAT_MONEY;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_MONEY_IMPORT))
                                swatSelect = SWAT_MONEY_IMPORT;

                        if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_MONEY_LINK))
                                swatSelect = SWAT_MONEY_LINK;

                        switch (swatSelect)
                        {
                        case SWAT_MONEY_IMPORT:
                                pSetup->pMoney->SetLink(FALSE);
                                break;

                        case SWAT_MONEY_LINK:
                                if (GET_ROAMING(pSetup->pOCX)->GetRoamingState() !=
RMS_OFF)
                                {

MessageBox(OCX(pSetup->pOCX)->GetInnerWindow(), GetRscString2(IDS_ROAMING_ERR_LINK),
GetRscString(IDS_ERROR), MB_OK | MB_ICONSTOP | MB_APPLMODAL);
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                                }
                                else
                                {
                                        pSetup->pMoney->SetLink(TRUE);
                                }
                                break;

                        default:
                                assert(FALSE);
                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                                break;
                        }

                        pSetup->swatSelect = swatSelect;
                        return TRUE;

                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
                        return TRUE;

                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                assert(pSetup->pMoney);

                CheckDlgButton(hwndDlg, IDC_ACCOUNT_MONEY_IMPORT, BST_CHECKED);

                if (!pSetup->pOCX->IsMoneyInstalled(MT_ANY))
                {
                        EnableControl(hwndDlg, IDC_ACCOUNT_MONEY_LINK, FALSE);
                }

                return(TRUE);
```

```
        }

        return(FALSE);
}

//--------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Money 2 Property Sheet.
//--------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizMoneyImportDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_COMMAND:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                switch(LOWORD(wParam))
                {
                case IDC_ACCOUNT_LIST:
                        if (HIWORD(wParam) == LBN_SELCHANGE)
                        {
                                CheckImportWizButtonStatus(pSetup, hwndDlg);
                        }
                        break;

                case IDC_FILE:
                        if (HIWORD(wParam) == EN_KILLFOCUS)
                        {
                                if (pSetup->fClosing == FALSE)
                                {
                                        // Do the default behavior
                                        MoneyAccountsDialogProc(hwndDlg, uMsg,
wParam, lParam);

                                        // Check the selection
                                        // CheckImportWizButtonStatus(hwndDlg);
                                }

                                // Don't pass this message along, because we already
did if necessary.
                                return(FALSE);
                        }
                        break;

                case IDC_BROWSE:
                        // Do the default behavior
                        MoneyAccountsDialogProc(hwndDlg, uMsg, wParam, lParam);

                        // Check the selection
                        CheckImportWizButtonStatus(pSetup, hwndDlg);

                        // Don't pass this message along, because we already did if
necessary.
                        return(FALSE);
                }
```

```
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        pSetup->fClosing = FALSE;
                        pSetup->fNext = FALSE;

                        PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);

                        pSetup->pMoney->SetReportError(FALSE);
                        pSetup->pMoney->FillAccountList(hwndDlg, NULL);


                        CheckImportWizButtonStatus(pSetup, hwndDlg);
                        break;

                case PSN_WIZNEXT:
                        pSetup->pMoney->SetReportError(TRUE);

                        if (pSetup->pMoney->GetAccountsSelected(hwndDlg))
                        {
                                pSetup->fClosing = TRUE;
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_MONEY_FINISH);
                        }
                        else
                        {
                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                        }

                        return TRUE;

                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_MONEY_1);
                        return TRUE;
                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                // ImportAccountsDialogProc handles most of the initialization
                lParam = (LPARAM)pSetup->pMoney;
                break;
        }

        return(MoneyAccountsDialogProc(hwndDlg, uMsg, wParam, lParam));
}

//----------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Money Finish Property Sheet.
//----------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizMoneyFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
```

```
        switch(uMsg)
        {
        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_MONEY_IMPORT);
                        return TRUE;
                }
                break;
        }

        return(AccountWizFinishDialogProc(hwndDlg, uMsg, wParam, lParam));
}

//----------------------------------------------------------------------------
-------
// Dialog proc to enable/disable the next button in response to various events
//----------------------------------------------------------------------------
-------
void CheckQuickenButtonStatus(SETUP_STRUCT *pSetup, HWND hwndDlg)
{
        CQuicken *pQuicken = pSetup->pQuicken;

        if (pQuicken->IsQReadInstalled())
                CheckImportWizButtonStatus(pSetup, hwndDlg);
        else
                PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
}

//----------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Quicken Property Sheet.
//----------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizQuickenDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_USER_QREAD_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                CheckQuickenButtonStatus(pSetup, hwndDlg);
                break;

        case WM_COMMAND:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                switch(LOWORD(wParam))
                {
                case IDC_ACCOUNT_LIST:
                        if (HIWORD(wParam) == LBN_SELCHANGE)
                        {
                                assert(GetDlgItem(hwndDlg, IDC_ACCOUNT_LIST) ==
```

```
(HWND)lParam);
                        CheckQuickenButtonStatus(pSetup, hwndDlg);
                    }
                    break;

                case IDC_BROWSE:
                    // Do the default behavior
                    QuickenAccountsDialogProc(hwndDlg, uMsg, wParam, lParam);

                    // Check the selection
                    CheckQuickenButtonStatus(pSetup, hwndDlg);

                    // Don't pass this message along, because we already did if
necessary.
                    return(FALSE);
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                    assert(pSetup->pQuicken);
                    pSetup->fNext = FALSE;
                    PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
                    pSetup->pQuicken->EnsureQReadDLL(NULL, hwndDlg);
                    CheckQuickenButtonStatus(pSetup, hwndDlg);
                    break;

                case PSN_WIZBACK:
                    SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
                    return TRUE;

                case PSN_WIZNEXT:
                    pSetup->pQuicken->SetReportError(TRUE);

                    if (pSetup->pQuicken->GetAccountsSelected(hwndDlg))
                    {
                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_QUICKEN_FINISH);
                    }
                    else
                    {
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                    }
                    return TRUE;
                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                // ImportAccountsDialogProc handles most of the initialization
                lParam = (LPARAM)pSetup->pQuicken;
                break;
        }

        return(QuickenAccountsDialogProc(hwndDlg, uMsg, wParam, lParam));
}
```

```
//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Quciken Finish Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizQuickenFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_QUICKEN);
                        return TRUE;

                }
                break;
        }

        return(AccountWizFinishDialogProc(hwndDlg, uMsg, wParam, lParam));
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard Master Password Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizMasterPasswordDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
        TCHAR szTitle[MAX_STRING];

        switch(uMsg)
        {
        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        if (pSetup->wNAWEntry == NAW_UPDATE_ACCOUNT)
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_NEXT);
                        else
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
                        break;

                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
```

```
                        return TRUE;

                case PSN_WIZNEXT:
                        if (ValidatePassword(hwndDlg, SEC_FLAG_VERIFY_LENGTH,
IDC_PASSWORD_NEW, IDC_PASSWORD_NEW_CONFIRM))
                        {

OCX(pSetup->pOCX)->SetLastMPChangeJulian(g_pDateToday->GetJulian());
                                GetDlgItemText(hwndDlg, IDC_PASSWORD_NEW,
pSetup->pOFX->GetPassword(), MAX_STRING);

OCX(pSetup->pOCX)->SetMasterPassword(pSetup->pOFX->GetPassword());
                                SetFirstOFXPropSheet(hwndDlg, pSetup);
                        }
                        else
                        {
                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                        }
                        return TRUE;
                }
                break;

        case WM_INITDIALOG:
                {
                        pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE
*)lParam)->lParam);
                        SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
                        SetDlgItemText(hwndDlg, IDC_PASSWORD_NEW,
pSetup->pOFX->GetPassword());
                        SetDlgItemText(hwndDlg, IDC_PASSWORD_NEW_CONFIRM,
pSetup->pOFX->GetPassword());

                        // Brand the title
                        if (GetWindowText(hwndDlg, szTitle, MAX_STRING))
                        {
                                OCX(pSetup->pOCX)->ApplyIPKBranding(szTitle, NULL,
MAX_STRING);
                                PropSheet_SetTitle(GetParent(hwndDlg), 0, szTitle);
                        }

                }
                return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------------
-------
// Activate controls for OFX import
//-------------------------------------------------------------------------------
-------
void AccountWizOFX1PSNActiveDlgProc(HWND hwndDlg)
{
        DWORD   dwFlags = 0;

        SETUP_STRUCT *pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
SETUPPROP_PROP_NAME);
        assert(pSetup);

        // Do we have the possibility of a next button? Check for a listview
selection.
        if (ListView_GetSelectedCount(GetDlgItem(hwndDlg, IDC_OFX_BROKER_LIST)) ==
```

```
1)
        {
                dwFlags |= PSWIZB_NEXT;
        }

        if ((pSetup->wNAWEntry != NAW_UPDATE_ACCOUNT)
                || ((pSetup->pOCX->GetSecurityLevel() != SECURITY_HIGH)
                && (FileHasEncryptedOFXData(pSetup->pOCX) == FALSE)))
        {
                dwFlags |= PSWIZB_BACK;
        }

        PropSheet_SetWizButtons(GetParent(hwndDlg), dwFlags);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX Property Sheet #1.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizOFX1DialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT            *pSetup = NULL;
        HWND                            hwndList;

        switch(uMsg)
        {
        case WM_USER_PARSER_DOWNLOAD:
                AccountWizOFX1PSNActiveDlgProc(hwndDlg);
                break;

        case WM_USER_BROKERLIST_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);

                hwndList = GetDlgItem(hwndDlg, IDC_OFX_BROKER_LIST);

                // Get bitmaps
                pSetup->pOFX->EnsureBrokerImages(hwndList);

                pSetup->pOFX->FillBrokerList(hwndList);

                EnableWindow(hwndList, TRUE);

                AccountWizOFX1PSNActiveDlgProc(hwndDlg);
                break;

        case WM_USER_BROKERIMAGE_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);

                hwndList = GetDlgItem(hwndDlg, IDC_OFX_BROKER_LIST);

                pSetup->pOFX->UpdateBrokerList(hwndList);

                pSetup->pOFX->EnsureBrokerImages(hwndList);
                break;

        case WM_PAINT:
                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;
```

```
        case WM_NOTIFY:
                if (wParam == IDC_OFX_BROKER_LIST)
                {
                        NM_LISTVIEW     *pLV = (NM_LISTVIEW *)lParam;

                        switch (pLV->hdr.code)
                        {
                        case NM_DBLCLK: // same as an OK
                                PropSheet_PressButton(GetParent(hwndDlg),
PSBTN_NEXT);
                                break;

                        case LVN_ITEMCHANGED:
                                if (pLV->uNewState & LVIS_SELECTED)
                                        AccountWizOFX1PSNActiveDlgProc(hwndDlg);
                                break;
                        }
                }
                else
                {
                        pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
SETUPPROP_PROP_NAME);
                        assert(pSetup);

                        switch (((NMHDR *)lParam)->code)
                        {
                        case PSN_QUERYCANCEL:

GET_ERROR(pSetup->pOCX)->DisplayNotification(GetParent(hwndDlg),

                        IDS_MORE_BROKERS_COMING,

                        IDS_PORTFOLIO_MANAGER,

                        MB_ICONEXCLAMATION, MB_OK);
                                return (FALSE);         // FALSE means accept the
cancel

                        case PSN_SETACTIVE:
                                assert(pSetup->pOFX);

                                // Read broker.txt.
                                pSetup->pOFX->EnsureBrokerList(NULL, hwndDlg);

                                pSetup->pOFX->EnsureOFXParser(NULL);

                                AccountWizOFX1PSNActiveDlgProc(hwndDlg);
                                break;

                        case PSN_WIZBACK:
                                if (FileHasEncryptedOFXData(pSetup->pOCX)
                                        || (pSetup->pOCX->GetSecurityLevel() ==
SECURITY_HIGH))
                                {
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
                                }
                                else
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_MASTER_PASSWORD);

                                return TRUE;
```

```
                        case PSN_WIZNEXT:
                                COFXSession     *pOFXSession;

                                hwndList = GetDlgItem(hwndDlg, IDC_OFX_BROKER_LIST);

                                if (ListView_GetSelectedCount(hwndList) == 0)
                                {
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                                        return(TRUE);
                                }

                                pOFXSession = pSetup->pOFX->GetCurSession();
                                if (pOFXSession !=
pSetup->pOFX->SetCurSessionToSelection(hwndList))
                                {
                                        pOFXSession = pSetup->pOFX->GetCurSession();

                                        // Broker selection changed, cancel the
previous profile download
                                        //  if one is in progress.
                                        if (pSetup->pOFXDownload)
                                        {
                                                // CancelDownload releases the
object, and our callback zeros out pOFXDownload

pSetup->pOFXDownload->CancelDownload();

ReleaseInterface(pSetup->pOFXDownload);
                                        }
                                }

                                if (pOFXSession->fNotGrouping())
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_NOGROUPING);
                                else
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_1);
                                return TRUE;
                        }
                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                if ((pSetup) && (pSetup->pOFX))
                        pSetup->pOFX->SetHWndParent(hwndDlg);

                return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------------
-------
// Change the dialog text and controls now that we have profile information
//-------------------------------------------------------------------------------
-------
void AccountWizOFXNavigationDlgProc(HWND hwndDlg, SETUP_STRUCT *pSetup, long
idcText1, long idcText2)
```

```
{
        COFXSession     *pOFXSession = pSetup->pOFX->GetCurSession();
        DWORD           dwWizButtons = 0;

        // if this is not being updated, or we are updating but we didn't know the
broker
        //  before we updated, we have a back button.
        if ((pSetup->wNAWEntry != NAW_UPDATE_ACCOUNT) || ((pSetup->wNAWModifier &
NAW_MODIFIER_NEW_ACCOUNT) != 0))
                dwWizButtons |= PSWIZB_BACK;

        // For the next button to be activated, both text fields must have text, and
the profile must be downloaded.
        if (pOFXSession && pOFXSession->GetProfile() && DoesEditBoxHaveText(hwndDlg,
idcText1) && DoesEditBoxHaveText(hwndDlg, idcText2))
                dwWizButtons |= PSWIZB_NEXT;

        PropSheet_SetWizButtons (GetParent(hwndDlg), dwWizButtons);
}

//---------------------------------------------------------------------------
-------
// Change the dialog text and controls with custom text from broker.txt
//---------------------------------------------------------------------------
-------
void AccountWizOFXCustomTextDlgProc(HWND hwndDlg)
{
        SETUP_STRUCT    *pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
SETUPPROP_PROP_NAME);
        COFXSession             *pOFXSession;
        OFX_BROKER              *pBroker;

        assert(pSetup);
        assert(pSetup->pOFX);

        pOFXSession = pSetup->pOFX->GetCurSession();
        pBroker = pSetup->pOFX->FindBroker(pOFXSession->GetBrokerID());

        // Prompts
        if (pBroker && pBroker->szUserIDPrompt[0])
                SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_USERID_CAPTION,
pBroker->szUserIDPrompt);
        else
                SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_USERID_CAPTION,
GetRscString(IDS_OFX_USERID_DEFAULT));

        if (pBroker && pBroker->szPasswordPrompt[0])
                SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_PASSWORD_CAPTION,
pBroker->szPasswordPrompt);
        else
                SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_PASSWORD_CAPTION,
GetRscString(IDS_OFX_PASSWORD_DEFAULT));

        if (pBroker && pBroker->szAccountPrompt[0])
                SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_NUMBER_CAPTION,
pBroker->szAccountPrompt);
        else
                SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_NUMBER_CAPTION,
GetRscString(IDS_OFX_ACCOUNT_NUMBER_DEFAULT));

        // Informational text items
        if (pBroker && pBroker->szSignupInfo[0])
                SetDlgItemText(hwndDlg, IDC_OFX_SIGNUP_INFO, pBroker->szSignupInfo);
```

```
        else
                SetDlgItemText(hwndDlg, IDC_OFX_SIGNUP_INFO,
GetRscString(IDS_OFX_SIGNUP_INFO_DEFAULT));

        if (pBroker && pBroker->szCustomInfo)
                SetDlgItemText(hwndDlg, IDC_OFX_CUSTOM_INFO, pBroker->szCustomInfo);

        if (pBroker && pBroker->szAccountInfo[0])
                SetDlgItemText(hwndDlg, IDC_OFX_ACCOUNT_INFO,
pBroker->szAccountInfo);
        else
                SetDlgItemText(hwndDlg, IDC_OFX_ACCOUNT_INFO,
GetRscString(IDS_OFX_ACCOUNT_INFO_DEFAULT));
}



//-------------------------------------------------------------------------------
-------
// Change the dialog text and controls now that we have profile information
//-------------------------------------------------------------------------------
-------
void AccountWizOFXProfileActiveDlgProc(HWND hwndDlg)
{
        SETUP_STRUCT    *pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
SETUPPROP_PROP_NAME);
        COFXSession             *pOFXSession = pSetup->pOFX->GetCurSession();

        assert(pSetup);
        assert(pOFXSession);
}


//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX No Account Grouping Property Sheet .
//-------------------------------------------------------------------------------
-------
BOOL GetBrokerProfileInformation(COFXSession *pOFXSession, HWND hwndDlg,
SETUP_STRUCT *pSetup)
{
        BOOL    fResult = FALSE;

        assert(pOFXSession);
        assert(hwndDlg);
        assert(pSetup);

        // Don't toss alerts if this dialog is running a download.
        //  Check by seeing if the progress bar is currently visible.
        if (ShowWindow(GetDlgItem(hwndDlg, IDC_DOWNLOAD_PROGRESS), SW_SHOWNA) == 0)
        // if (!IsWindowVisible(GetDlgItem(hwndDlg, IDC_DOWNLOAD_PROGRESS)))
        {
                COFXProfile     *pProfile = new COFXProfile(pSetup->pOCX, hwndDlg,
pSetup->pOFX->GetOFXCom());

                pOFXSession->SetLParam((LPARAM)hwndDlg);
                fResult = pProfile->InitiateProfile(pOFXSession);

                // We should have cancelled any pending profile download already.
                assert(pSetup->pOFXDownload == NULL);
                pSetup->pOFXDownload = pOFXSession->GetDownload();
                pSetup->pOFXDownload->AddRef();
        }
```

```
        return(fResult);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX No Account Grouping Property Sheet .
//-------------------------------------------------------------------------------
-------
BOOL AccountWizBrokerListDlgProc(HWND hwndDlg, SETUP_STRUCT *pSetup, long idsTitle,
long idcText1, long idcText2)
{
        BOOL                    fSuccess = FALSE;
        COFXSession             *pOFXSession;

        assert(pSetup);

        pOFXSession = pSetup->pOFX->GetCurSession();

        if (pOFXSession)
        {
                char    szTitle[MAX_STRING];

                AccountWizOFXNavigationDlgProc(hwndDlg, pSetup, idcText1, idcText2);

                if (!pOFXSession->GetProfile())
                {
                        fSuccess = GetBrokerProfileInformation(pOFXSession, hwndDlg,
pSetup);
                }
                else
                {
                        fSuccess = TRUE;
                        AccountWizOFXProfileActiveDlgProc(hwndDlg);
                }

                // Set the prop sheet title
                wsprintf(szTitle, GetRscString(idsTitle),
pOFXSession->GetServerName());
                PropSheet_SetTitle(GetParent(hwndDlg), 0, szTitle);
        }

        return(fSuccess);
}


//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX No Account Grouping Property Sheet .
//-------------------------------------------------------------------------------
-------
COFXSession *SetCurSessionToIPKBroker(SETUP_STRUCT *pSetup)
{
        COFXSession     *pOFXSession = NULL;

        if (pSetup->pOCX->GetIPKBrokerID() != NULL_BROKER)
        {
                OFX_BROKER *pBroker =
pSetup->pOFX->FindBroker(pSetup->pOCX->GetIPKBrokerID());
                assert(pBroker);

                pOFXSession = pSetup->pOFX->SetCurSessionToBroker(pBroker);
        }
```

```
        return(pOFXSession);
}


//----------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX No Account Grouping Property Sheet .
//----------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizOFXNoGroupingDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;

        switch(uMsg)
        {
        case WM_USER_SETFOCUS:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
                break;

        case WM_USER_PARSER_DOWNLOAD:
                break;

        case WM_USER_BROKERLIST_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);

                if (pSetup->pOCX->IsIPK())
                        SetCurSessionToIPKBroker(pSetup);

                {
                        COFXSession *pOFXSession = pSetup->pOFX->GetCurSession();
                        OFX_BROKER *pBroker =
pSetup->pOFX->FindBroker(pOFXSession->GetBrokerID());

                        // Get bitmaps
                        pSetup->pOFX->DownloadBrokerImage(pBroker);
                }

                AccountWizBrokerListDlgProc(hwndDlg, pSetup, IDS_OFX_ACCOUNT_TITLE,
IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
                break;

        case WM_USER_OFX_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                ReleaseInterface(pSetup->pOFXDownload);

                AccountWizOFXNavigationDlgProc(hwndDlg, pSetup,
IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
                AccountWizOFXProfileActiveDlgProc(hwndDlg);
                break;

        case WM_USER_BROKERIMAGE_DOWNLOAD:
                // intentionally repaint bitmap.
        case WM_PAINT:
                if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
                        PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);

                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
```

```
                break;

        case WM_COMMAND:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                switch(HIWORD(wParam))
                {
                case EN_CHANGE:
                        AccountWizOFXNavigationDlgProc(hwndDlg, pSetup,
IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
                        break;
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        assert(pSetup->pOFX);

                        AccountWizOFXCustomTextDlgProc(hwndDlg);

                        pSetup->pOFX->EnsureOFXParser(NULL);

                        if (pSetup->pOFX->EnsureBrokerList(NULL, hwndDlg))
                        {
                                // We have the list already, do normal activation.
                                AccountWizBrokerListDlgProc(hwndDlg, pSetup,
IDS_OFX_ACCOUNT_TITLE, IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
                        }
                        break;

                case PSN_WIZBACK:
                        // Grab dialog info before we leave in case we come back
                        pSetup->pOFX->GetAccountData(hwndDlg, 0,
IDC_ACCOUNT_OFX_PASSWORD, IDC_ACCOUNT_OFX_NUMBER);

                        // IPK skips over the dialog to select a broker
                        if (pSetup->pOCX->GetIPKBrokerID() == NULL_BROKER)
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_1);
                        else
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
                        return(TRUE);

                case PSN_WIZNEXT:
                        pSetup->pOFX->GetAccountData(hwndDlg,
IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD, IDC_ACCOUNT_OFX_NUMBER);

                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_FINISH);
                        return TRUE;
                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                assert(pSetup->pOFX);
                if ((pSetup) && (pSetup->pOFX))
```

```
                {
                        pSetup->pOFX->SetAccountData(hwndDlg, 0,
IDC_ACCOUNT_OFX_PASSWORD, IDC_ACCOUNT_OFX_NUMBER);
                        pSetup->pOFX->SetHWndParent(hwndDlg);
                }

                return(TRUE);
        }

        return(FALSE);
}


//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX Grouping 1 Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizOFXGrouping1DialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
        COFXSession             *pOFXSession;

        switch(uMsg)
        {
        case WM_USER_SETFOCUS:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
                break;

        case WM_USER_PARSER_DOWNLOAD:
                break;

        case WM_USER_BROKERLIST_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);

                if (pSetup->pOCX->IsIPK())
                        SetCurSessionToIPKBroker(pSetup);

                {
                    COFXSession *pOFXSession = pSetup->pOFX->GetCurSession();
                        OFX_BROKER *pBroker;

                        assert(pOFXSession);

                        pSetup->pOFX->EnsureSessionParams(pOFXSession);

                        pBroker =
pSetup->pOFX->FindBroker(pOFXSession->GetBrokerID());
                        assert(pBroker);

                        if (pOFXSession->fNotGrouping())
                        {
                                // Need to switch to NoGroupingDialog. This can
happen through the
                                //  Account details dialog if the user clicks the
setup button
                                //  to access OFX account information after setting
up a
                                //  Non-grouping OFX account.
                                PropSheet_SetCurSelByID(GetParent(hwndDlg),
```

```
IDD_ACCOUNT_OFX_NOGROUPING);
                        }
                        else
                        {
                                // Get bitmaps
                                pSetup->pOFX->DownloadBrokerImage(pBroker);

                                AccountWizBrokerListDlgProc(hwndDlg, pSetup,
IDS_OFX_ACCOUNT_TITLE, IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
                        }
                }

                break;

        case WM_USER_OFX_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                ReleaseInterface(pSetup->pOFXDownload);

                AccountWizOFXNavigationDlgProc(hwndDlg, pSetup,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD);
                AccountWizOFXProfileActiveDlgProc(hwndDlg);
                break;

        case WM_USER_BROKERIMAGE_DOWNLOAD:
                // intentionally repaint bitmap.
        case WM_PAINT:
                if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
                        PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);

                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_COMMAND:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                switch(HIWORD(wParam))
                {
                case EN_CHANGE:
                        AccountWizOFXNavigationDlgProc(hwndDlg, pSetup,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD);
                        break;
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        assert(pSetup->pOFX);

                        AccountWizOFXCustomTextDlgProc(hwndDlg);

                        pSetup->pOFX->EnsureOFXParser(NULL);

                        if (pSetup->pOFX->EnsureBrokerList(NULL, hwndDlg))
                        {
                                // We have the list already, do normal activation.
                                AccountWizBrokerListDlgProc(hwndDlg, pSetup,
IDS_OFX_USER_TITLE, IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD);
                        }
```

```
                        break;

                case PSN_WIZBACK:
                        // Save data
                        pSetup->pOFX->GetAccountData(hwndDlg,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD, 0);

                        // IPK skips over the dialog to select a broker
                        if (pSetup->pOCX->GetIPKBrokerID() == NULL_BROKER)
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_1);
                        else
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);

                        return(TRUE);

                case PSN_WIZNEXT:
                        pSetup->pOFX->GetAccountData(hwndDlg,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD, 0);

                        pOFXSession = pSetup->pOFX->GetCurSession();
                        assert(pOFXSession);

                        {
                                COFXSignup      *pSignup = new
COFXSignup(pSetup->pOCX, hwndDlg, pSetup->pOFX->GetOFXCom());

                                if (pSignup->ValidateProfile(pOFXSession))
                                {
                                        // TODO: Need to check if the FI supports
ACCTINFO.
                                        if (pSignup->GetAvailAccts())
                                                SetWindowLong(hwndDlg,
DWL_MSGRESULT, IDD_ACCOUNT_OFX_GROUPING_3);
                                        else
                                                SetWindowLong(hwndDlg,
DWL_MSGRESULT, IDD_ACCOUNT_OFX_GROUPING_2);
                                }
                                else
                                {
                                        assert(FALSE);  // Broker profile is bad!
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_2);
                                }

                                DeleteInterface(pSignup);
                        }

                        return(TRUE);

                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                assert(pSetup);
                if (pSetup && pSetup->pOFX)
                {
                        pSetup->pOFX->SetHWndParent(hwndDlg);
```

```
                        pSetup->pOFX->SetAccountData(hwndDlg,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD, 0);
                }

                return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX Grouping 2 Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizOFXGrouping2DialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
        COFXSession             *pOFXSession;

        switch(uMsg)
        {
        case WM_USER_SETFOCUS:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
                break;

        case WM_PAINT:
                if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
                        PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);

                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_COMMAND:
                switch(HIWORD(wParam))
                {
                case EN_CHANGE:
                        if (DoesEditBoxHaveText(hwndDlg, IDC_ACCOUNT_OFX_NUMBER))
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
                        else
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
                        break;
                }
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        AccountWizOFXCustomTextDlgProc(hwndDlg);

                        if (DoesEditBoxHaveText(hwndDlg, IDC_ACCOUNT_OFX_NUMBER))
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
```

```
                                        setupwiz
                        else
                                PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);

                        pOFXSession = pSetup->pOFX->GetCurSession();

                        if (pOFXSession)
                        {
                                TCHAR   szInfo[MAX_STRING];

                                wsprintf(szInfo,
GetRscString(IDS_OFX_ACCOUNT_TITLE), pOFXSession->GetServerName());
                                PropSheet_SetTitle(GetParent(hwndDlg), 0, szInfo);
                        }
                        break;

                case PSN_WIZBACK:
                        pSetup->pOFX->GetAccountData(hwndDlg, 0, 0,
IDC_ACCOUNT_OFX_NUMBER);

                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_1);
                        return(TRUE);

                case PSN_WIZNEXT:
                        pSetup->pOFX->GetAccountData(hwndDlg, 0, 0,
IDC_ACCOUNT_OFX_NUMBER);

                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_FINISH);
                        return(TRUE);

                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

                if (pSetup && pSetup->pOFX)
                {
                        pSetup->pOFX->SetHWndParent(hwndDlg);

                        // Prefill dialog items if we have account data...
                        pSetup->pOFX->SetAccountData(hwndDlg, 0, 0,
IDC_ACCOUNT_OFX_NUMBER);
                }

                return(TRUE);
        }

        return(FALSE);
}


//-------------------------------------------------------------------------
-------
// Execute an ACCTINFO OFX download to get the list of accounts.
//-------------------------------------------------------------------------
-------
BOOL GetBrokerAccountInformation(COFXSession *pOFXSession, HWND hwndDlg,
SETUP_STRUCT *pSetup)
{
```

```
        BOOL            fResult = FALSE;

        assert(pOFXSession);
        assert(hwndDlg);
        assert(pSetup);

        // Don't toss alerts if this dialog is running a download.
        //  Check by seeing if the progress bar is currently visible.
        if (ShowWindow(GetDlgItem(hwndDlg, IDC_DOWNLOAD_PROGRESS), SW_SHOWNA) == 0)
        // if (!IsWindowVisible(GetDlgItem(hwndDlg, IDC_DOWNLOAD_PROGRESS)))
        {
                COFXSignup      *pSignup = new COFXSignup(pSetup->pOCX, hwndDlg,
pSetup->pOFX->GetOFXCom());

                // Clear the list
                SendDlgItemMessage(hwndDlg, IDC_ACCOUNT_LIST, LB_RESETCONTENT, 0,
0);

                // Start Account Info download
                pOFXSession->SetLParam((LPARAM)hwndDlg);
                fResult = pSignup->InitiateSignup(pOFXSession);

                // We should have cancelled any pending profile download already.
                assert(pSetup->pOFXDownload == NULL);
                pSetup->pOFXDownload = pOFXSession->GetDownload();
                pSetup->pOFXDownload->AddRef();
        }

        return(fResult);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX Grouping 3 Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizOFXGrouping3DialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
        COFXSession             *pOFXSession;

        switch(uMsg)
        {
        case WM_USER_SETFOCUS:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
                break;

        case WM_USER_OFX_DOWNLOAD:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                ReleaseInterface(pSetup->pOFXDownload);

                // Populate the list with the accounts.
                if (pSetup->pOFX->FillAccountList(hwndDlg, NULL))
                {
                        // Update the navigation elements.
                        PropSheet_SetWizButtons(GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
                }

                break;
```

```
        case WM_PAINT:
                if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
                        PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);

                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
                break;

        case WM_NOTIFY:
                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
                assert(pSetup);
                switch (((NMHDR *)lParam)->code)
                {
                case PSN_SETACTIVE:
                        pOFXSession = pSetup->pOFX->GetCurSession();
                        assert(pOFXSession);

                        if (pOFXSession)
                        {
                                TCHAR   szTitle[MAX_STRING];

                                // If HasAccounts returns TRUE, we have already
downloaded the accounts
                                //  for this session.
                                if (pOFXSession->GetHasAccounts() == FALSE)
                                {
                                        PropSheet_SetWizButtons(GetParent(hwndDlg),
PSWIZB_BACK);

                                        GetBrokerAccountInformation(pOFXSession,
hwndDlg, pSetup);
                                }
                                else
                                {
                                        PropSheet_SetWizButtons(GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
                                }

                                // Set the propsheet title
                                wsprintf(szTitle,
GetRscString(IDS_OFX_SELECT_TITLE), pOFXSession->GetServerName());
                                PropSheet_SetTitle(GetParent(hwndDlg), 0, szTitle);
                        }
                        break;

                case PSN_WIZBACK:
                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_1);
                        return(TRUE);

                case PSN_WIZNEXT:
                        pSetup->pOFX->SetReportError(TRUE);

                        if (pSetup->pOFX->GetAccountsSelected(hwndDlg))
                        {
                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_FINISH);
                        }
                        else
                        {
                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
                        }
```

```
                return(TRUE);

            }
            break;

        case WM_INITDIALOG:
            pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
            SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);

            if ((pSetup) && (pSetup->pOFX))
            {
                pSetup->pOFX->SetHWndParent(hwndDlg);

                if (pSetup->pOFX->GetCurAccount() == NULL)
                {
                    HWND    hwndList;

                    hwndList = GetDlgItem(hwndDlg, IDC_ACCOUNT_LIST);

                    ULONG ulStyle = GetWindowLong(hwndList, GWL_STYLE);
                    ulStyle |= LBS_MULTIPLESEL;
                    SetWindowLong(hwndList, GWL_STYLE, ulStyle);
                }
            }

            return(TRUE);
        }

        return(FALSE);
}

//-------------------------------------------------------------------------------
-------
// Dialog proc for the Account Wizard OFX Finish Property Sheet.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK AccountWizOFXFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
        SETUP_STRUCT    *pSetup = NULL;
        COFXSession             *pOFXSession;

        switch(uMsg)
        {
        case WM_PAINT:
            if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
                PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);
            break;

        case WM_NOTIFY:
            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
            assert(pSetup);
            switch (((NMHDR *)lParam)->code)
            {
            case PSN_WIZBACK:
                pOFXSession = pSetup->pOFX->GetCurSession();

                if (pOFXSession->fNotGrouping())
                    SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_NOGROUPING);
                else
                {
```

```
                                        setupwiz
                                if (pOFXSession->GetHasAccounts())
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_3);
                                else
                                        SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_2);
                        }
                        return TRUE;

                case PSN_WIZFINISH:
                        pOFXSession = pSetup->pOFX->GetCurSession();

                        if (pOFXSession->GetHasAccounts())
                        {
                                pSetup->pOFX->TrimAccountPlex();
                        }

                        // For Grouping2 case, this flag was never set.
                        pOFXSession->PutHasAccounts(TRUE);

                        break;
                }
                break;

        case WM_INITDIALOG:
                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
                if (pSetup && pSetup->pOCX)
                        pSetup->pOCX->ApplyIPKBranding(hwndDlg,
IDC_OFX_FINISH_TEXT1);
                break;

        }

        return(AccountWizFinishDialogProc(hwndDlg, uMsg, wParam, lParam));
}


//-------------------------------------------------------------------------------
-------
// The beginnings of Investor's tip wizard...
//
//      Dialog proc for Tip Wizard dialog. Returns TRUE if you should turn off the
tip, FALSE
// if the tip should appear again next time.
//-------------------------------------------------------------------------------
-------
BOOL CALLBACK TipDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
        HBITMAP hBmpTip = NULL;
        RECT rect;

        switch(uMsg)
        {
        case WM_PAINT:
                {
                        int iTip = (int)GetProp(hwndDlg, TIP_ID_PROP);
                        GetCtlRect(GetDlgItem(hwndDlg, IDC_BUY_BITMAP), hwndDlg,
&rect); // client coordinates
                        PaintDlgBmp(hwndDlg, rgTips[iTip].residBitmap, NULL,
rect.left, rect.top,
                                rect.right - rect.left, rect.bottom - rect.top,
TRUE);
                }
```

```
                break;

        case WM_COMMAND:
                switch(LOWORD(wParam))
                {
                case IDOK:
                        EndDialog(hwndDlg, IsDlgButtonChecked(hwndDlg,
IDC_TIP_NOT_AGAIN));
                        break;
                }
                break;

        case WM_INITDIALOG:
                SetDlgItemText(hwndDlg, IDC_TIP_TEXT,
GetRscString(rgTips[lParam].residPrompt));
                SetProp(hwndDlg, TIP_ID_PROP, (HANDLE)lParam);
                break;
        }

        return(FALSE);
}


//-------------------------------------------------------------------------------
-------
// SYMBOLPRICE compare function
//-------------------------------------------------------------------------------
-------
SGN SPCompare(SYMBOLPRICE *psp1, SYMBOLPRICE *psp2)
{
        return ((SGN)lstrcmp(psp1->szSymbol, psp2->szSymbol));
}


//-------------------------------------------------------------------------------
-------
// Add Watch Position
//-------------------------------------------------------------------------------
-------
BOOL AddWatchPosition(CAccount *pAccount, TCHAR *pszTicker,
                CDate *pDate, double dblUnits, double dblPrice)
{
        CPosition               *pPosition = NULL;
        CTransaction    *pTrans = NULL;
        CURID                   curid = CUR_US;

        assert(*pszTicker != 0);        // gotta have a ticker, even if it is "none"

        pPosition = GET_DATA(pAccount->GetOCX())->AddPosition(pszTicker, TRUE,
pAccount->GetAccountNumber());

        if (pPosition)
        {
                GET_ROAMING(pAccount->GetOCX())->AddPositionToBatch(pPosition);

                // Add currency...
                curid = InferCurIdFromTicker(pszTicker, TRUE);
                pPosition->SetCurrencyID(curid);

                // Make sure we get quotes for this rate in the future...
                OCX(pAccount->GetOCX())->AddCurrencyRate(curid);

                // Apply currency scaling
```

```
                dblPrice *= g_rgCurrency[curid].dblScaling;

                pTrans = pPosition->AddTransaction(TT_BUY, TO_MANUAL, pDate,
dblUnits, dblPrice, 0);

                if (pTrans)

GET_ROAMING(pAccount->GetOCX())->AddTransactionToBatch(pTrans, TRUE);
        }

        return (pTrans != NULL);
}


//-------------------------------------------------------------------------------
-------
// SetupWatchAccount
//
//      Create the initial positions in the model portfolio...
//-------------------------------------------------------------------------------
-------
BOOL SetupWatchAccount(SETUP_STRUCT *pSetup, CAccount *pAccount)
{
        CPricePlex      *pPrices = NULL;
        SYMBOLPRICE     sp = {0};
        SYMBOLPRICE     *pSPCur, *pSPMac;
        CLISTATUS       status;
        TCHAR           *pszTickers = pSetup->pAds->szSymbols;
        TCHAR           szTicker[MAX_TICKER_NAME];
        int                     iSymbols;
        int                     iDefWatchQuant =
pSetup->pOCX->GetDefaultWatchQuant();
        CDate           date;
        SYSTEMTIME      st;
        BOOL            fNoPrice = FALSE;

        GetLocalTime(&st);
        date.SetDate(&st);

        while (pszTickers != NULL)
        {
                pszTickers = GetNextTerm(pszTickers, SYMBREAKPTR, TRUE, szTicker,
MAX_TICKER_NAME, NULL);

                if (*szTicker != 0)
                {
                        if (pPrices == NULL)
                                pPrices = new CPricePlex(5, 5, &SPCompare);

                        lstrcpyn(sp.szSymbol, szTicker, MAX_TICKER_NAME);
                        sp.dblPrice = 0;

                        pPrices->AddUniqueItem(&sp);
                }
        }

        if ((pPrices == NULL) || (pPrices->GetCount() == 0))
                return(FALSE);

        if (pSetup->pAds->fModel)
        {
                CDate   *pDate;
```

```
		if (!SetDateToString(&date, pSetup->pAds->szMODate, NULL))
			return(FALSE);

		// Get the closing price at the transaction date, if a valid date
has been entered.
		// If the transaction date is today, use the current price instead.
		if (date.Compare(*g_pDateToday) == 0)
			pDate = NULL;
		else
			pDate = &date;

		pSetup->pOCX->InitWosaStatusStruct(&status, qrsPortBrief);

		LookupPrices(pPrices, pDate, pSetup->pOCX->m_spClientSite, &status);

		iSymbols = pPrices->GetCount();

		if (iSymbols == 0)
			return(FALSE);
	}
	else
	{
		pSetup->pAds->mo = MO_NONE;
	}

	// If roaming, a batch must already be started when this code runs.

	switch (pSetup->pAds->mo)
	{
		case MO_NONE:
			// Just add the symbols - ordinary watch account
			FORPLEX(pSPCur, pSPMac, *pPrices)
			{
				AddWatchPosition(pAccount, pSPCur->szSymbol, &date,
iDefWatchQuant, 0);
			}
			break;

		case MO_DOLLARS:
			// $10000 per symbol.
			FORPLEX(pSPCur, pSPMac, *pPrices)
			{
				double p = pSPCur->dblPrice;
				if (p!=0)
					AddWatchPosition(pAccount, pSPCur->szSymbol,
&date, 10000/p, p);
				else
					fNoPrice = TRUE;
			}
			break;

		case MO_SHARES:
			// 100 shares each
			FORPLEX(pSPCur, pSPMac, *pPrices)
			{
				AddWatchPosition(pAccount, pSPCur->szSymbol, &date,
100, pSPCur->dblPrice);

				if (CloseEnough(pSPCur->dblPrice, 0, 0.000001))
					fNoPrice = TRUE;
			}
			break;
```

```
                case MO_TOTAL:
                        // pSetup->pAds->dblMOTotal for entire account
                        FORPLEX(pSPCur, pSPMac, *pPrices)
                        {
                                double p = pSPCur->dblPrice;
                                if (p!=0)
                                        AddWatchPosition(pAccount, pSPCur->szSymbol,
&date, pSetup->pAds->dblMOTotal/p/iSymbols, p);
                                else
                                        fNoPrice = TRUE;
                        }
                        break;

                default:
                        assert(FALSE);
        }

        GET_DATA(pSetup->pOCX)->PostAllTransactions(POST_SUMS);
        GET_LIST(pSetup->pOCX)->Refresh(TRUE);
        OCX(pSetup->pOCX)->LaunchUpdate();

        if (fNoPrice)
        {
                TCHAR   szError[MAX_ERROR_STRING];
                BOOL    fFirst = TRUE;

                GetRscStringToBuffer(IDS_ERR_MODEL_PRICE, szError);

                FORPLEX(pSPCur, pSPMac, *pPrices)
                {
                        if (CloseEnough(pSPCur->dblPrice, 0.0, 0.000001))
                        {
                                if (fFirst)
                                        fFirst = FALSE;
                                else
                                        my_strcatn(szError, ", ", MAX_ERROR_STRING);

                                my_strcatn(szError, pSPCur->szSymbol,
MAX_ERROR_STRING);
                        }
                }

                my_strcatn(szError, ". ", MAX_ERROR_STRING);

                if ((pSetup->pAds->mo == MO_DOLLARS) || (pSetup->pAds->mo ==
MO_TOTAL))
                {
                        my_strcatn(szError, GetRscString(IDS_ERR_MODEL_CALCULATION),
MAX_ERROR_STRING);
                }

                MessageBox(OCX(pSetup->pOCX)->GetInnerWindow(), szError,
GetRscString(IDS_ERROR), MB_OK | MB_ICONSTOP | MB_APPLMODAL);
        }

        // cleanup the plex
        if (pPrices)
                delete pPrices;

        return(TRUE);
}

//--------------------------------------------------------------------------
```

```
-------
// Dialog proc for the new account wizard.
//--------------------------------------------------------------------------
-------
void InitAccountWizardPSP(CPortfolioControl *pOCX, SETUP_STRUCT *pSetup,
                                                                INVPROPSHEETPAGE
psp[SETUP_WIZARD_PROPSHEETS])
{
        my_memset(psp, 0, sizeof(INVPROPSHEETPAGE) * SETUP_WIZARD_PROPSHEETS);

        // Setup Wizard
        psp[SWP_ACCOUNT_FTUE].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_FTUE].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_FTUE].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_FTUE);
        psp[SWP_ACCOUNT_FTUE].pfnDlgProc = (DLGPROC)AccountWizFTUEDialogProc;
        psp[SWP_ACCOUNT_FTUE].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_FTUE].lParam = (long)pSetup;

        // New Account
        psp[SWP_ACCOUNT_NEW].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_NEW].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_NEW].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_NEW);
        psp[SWP_ACCOUNT_NEW].pfnDlgProc = (DLGPROC)AccountWizNewDialogProc;
        psp[SWP_ACCOUNT_NEW].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_NEW].lParam = (long)pSetup;

        // Regular Accounts
        psp[SWP_ACCOUNT_REGULAR].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_REGULAR].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_REGULAR].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_REGULAR);
        psp[SWP_ACCOUNT_REGULAR].pfnDlgProc = (DLGPROC)AccountWizRegularDialogProc;
        psp[SWP_ACCOUNT_REGULAR].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_REGULAR].lParam = (long)pSetup;

        // Watch Accounts
        psp[SWP_ACCOUNT_WATCH].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_WATCH].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_WATCH].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_WATCH);
        psp[SWP_ACCOUNT_WATCH].pfnDlgProc = (DLGPROC)AccountWizWatchDialogProc;
        psp[SWP_ACCOUNT_WATCH].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_WATCH].lParam = (long)pSetup;

        // Import Money, Quicken, Yahoo!, OFX
        psp[SWP_ACCOUNT_EXISTING_IMPORT].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_EXISTING_IMPORT].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_EXISTING_IMPORT].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_IMPORT_EXISTING);
        psp[SWP_ACCOUNT_EXISTING_IMPORT].pfnDlgProc =
(DLGPROC)AccountWizImportExistingDialogProc;
        psp[SWP_ACCOUNT_EXISTING_IMPORT].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_EXISTING_IMPORT].lParam = (long)pSetup;

        // Import Web
        psp[SWP_ACCOUNT_WEB].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_WEB].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_WEB].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_WEB);
        psp[SWP_ACCOUNT_WEB].pfnDlgProc = (DLGPROC)AccountWizWebDialogProc;
        psp[SWP_ACCOUNT_WEB].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_WEB].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_WEB_LIST].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_WEB_LIST].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_WEB_LIST].pszTemplate =
```

```
MAKEINTRESOURCE(IDD_ACCOUNT_WEB_LIST);
        psp[SWP_ACCOUNT_WEB_LIST].pfnDlgProc = (DLGPROC)AccountWizWebListDialogProc;
        psp[SWP_ACCOUNT_WEB_LIST].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_WEB_LIST].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_WEB_FINISH].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_WEB_FINISH].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_WEB_FINISH].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_WEB_FINISH);
        psp[SWP_ACCOUNT_WEB_FINISH].pfnDlgProc =
(DLGPROC)AccountWizWebFinishDialogProc;
        psp[SWP_ACCOUNT_WEB_FINISH].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_WEB_FINISH].lParam = (long)pSetup;

        // Money Accounts
        psp[SWP_ACCOUNT_MONEY_1].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_MONEY_1].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_MONEY_1].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_MONEY_1);
        psp[SWP_ACCOUNT_MONEY_1].pfnDlgProc = (DLGPROC)AccountWizMoney1DialogProc;
        psp[SWP_ACCOUNT_MONEY_1].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_MONEY_1].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_MONEY_IMPORT].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_MONEY_IMPORT].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_MONEY_IMPORT].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_MONEY_IMPORT);
        psp[SWP_ACCOUNT_MONEY_IMPORT].pfnDlgProc =
(DLGPROC)AccountWizMoneyImportDialogProc;
        psp[SWP_ACCOUNT_MONEY_IMPORT].pszTitle = 0;
        psp[SWP_ACCOUNT_MONEY_IMPORT].dwFlags = PSP_USETITLE;
        psp[SWP_ACCOUNT_MONEY_IMPORT].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_MONEY_FINISH].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_MONEY_FINISH].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_MONEY_FINISH].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_MONEY_FINISH);
        psp[SWP_ACCOUNT_MONEY_FINISH].pfnDlgProc =
(DLGPROC)AccountWizMoneyFinishDialogProc;
        psp[SWP_ACCOUNT_MONEY_FINISH].pszTitle = 0;
        psp[SWP_ACCOUNT_MONEY_FINISH].lParam = (long)pSetup;

        // Quicken Accounts
        psp[SWP_ACCOUNT_QUICKEN].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_QUICKEN].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_QUICKEN].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_QUICKEN);
        psp[SWP_ACCOUNT_QUICKEN].pfnDlgProc = (DLGPROC)AccountWizQuickenDialogProc;
        psp[SWP_ACCOUNT_QUICKEN].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_QUICKEN].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_QUICKEN_FINISH].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_QUICKEN_FINISH].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_QUICKEN_FINISH].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_QUICKEN_FINISH);
        psp[SWP_ACCOUNT_QUICKEN_FINISH].pfnDlgProc =
(DLGPROC)AccountWizQuickenFinishDialogProc;
        psp[SWP_ACCOUNT_QUICKEN_FINISH].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_QUICKEN_FINISH].lParam = (long)pSetup;

        // Master Password
        psp[SWP_ACCOUNT_MASTER_PASSWORD].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_MASTER_PASSWORD].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_MASTER_PASSWORD].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_MASTER_PASSWORD);
```

```
        psp[SWP_ACCOUNT_MASTER_PASSWORD].pfnDlgProc =
(DLGPROC)AccountWizMasterPasswordDialogProc;
        psp[SWP_ACCOUNT_MASTER_PASSWORD].pszTitle = 0;
        psp[SWP_ACCOUNT_MASTER_PASSWORD].dwFlags = PSP_USETITLE;
        psp[SWP_ACCOUNT_MASTER_PASSWORD].lParam = (long)pSetup;

        // OFX Accounts
        psp[SWP_ACCOUNT_OFX_1].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_OFX_1].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_OFX_1].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_OFX_1);
        psp[SWP_ACCOUNT_OFX_1].pfnDlgProc = (DLGPROC)AccountWizOFX1DialogProc;
        psp[SWP_ACCOUNT_OFX_1].pszTitle = 0;
        psp[SWP_ACCOUNT_OFX_1].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_OFX_NOGROUPING].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_OFX_NOGROUPING].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_OFX_NOGROUPING].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_NOGROUPING);
        psp[SWP_ACCOUNT_OFX_NOGROUPING].pfnDlgProc =
(DLGPROC)AccountWizOFXNoGroupingDialogProc;
        psp[SWP_ACCOUNT_OFX_NOGROUPING].pszTitle = 0;
        psp[SWP_ACCOUNT_OFX_NOGROUPING].dwFlags = PSP_USETITLE;
        psp[SWP_ACCOUNT_OFX_NOGROUPING].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_OFX_GROUPING_1].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_OFX_GROUPING_1].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_OFX_GROUPING_1].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_GROUPING_1);
        psp[SWP_ACCOUNT_OFX_GROUPING_1].pfnDlgProc =
(DLGPROC)AccountWizOFXGrouping1DialogProc;
        psp[SWP_ACCOUNT_OFX_GROUPING_1].pszTitle = 0;
        psp[SWP_ACCOUNT_OFX_GROUPING_1].dwFlags = PSP_USETITLE;
        psp[SWP_ACCOUNT_OFX_GROUPING_1].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_OFX_GROUPING_2].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_OFX_GROUPING_2].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_OFX_GROUPING_2].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_GROUPING_2);
        psp[SWP_ACCOUNT_OFX_GROUPING_2].pfnDlgProc =
(DLGPROC)AccountWizOFXGrouping2DialogProc;
        psp[SWP_ACCOUNT_OFX_GROUPING_2].pszTitle = 0;
        psp[SWP_ACCOUNT_OFX_GROUPING_2].dwFlags = PSP_USETITLE;
        psp[SWP_ACCOUNT_OFX_GROUPING_2].lParam = (long)pSetup;

        psp[SWP_ACCOUNT_OFX_GROUPING_3].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_OFX_GROUPING_3].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_OFX_GROUPING_3].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_GROUPING_3);
        psp[SWP_ACCOUNT_OFX_GROUPING_3].pfnDlgProc =
(DLGPROC)AccountWizOFXGrouping3DialogProc;
        psp[SWP_ACCOUNT_OFX_GROUPING_3].pszTitle = 0;
        psp[SWP_ACCOUNT_OFX_GROUPING_3].dwFlags = PSP_USETITLE;
        psp[SWP_ACCOUNT_OFX_GROUPING_3].lParam = (long)pSetup;

        // End
        psp[SWP_ACCOUNT_OFX_FINISH].dwSize = sizeof(INVPROPSHEETPAGE);
        psp[SWP_ACCOUNT_OFX_FINISH].hInstance = OCX(pOCX)->GetInstance();
        psp[SWP_ACCOUNT_OFX_FINISH].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_FINISH);
        psp[SWP_ACCOUNT_OFX_FINISH].pfnDlgProc =
(DLGPROC)AccountWizOFXFinishDialogProc;
        psp[SWP_ACCOUNT_OFX_FINISH].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
        psp[SWP_ACCOUNT_OFX_FINISH].lParam = (long)pSetup;
```

```
}

//--------------------------------------------------------------------------
-------
// The beginnings of Investor's tip wizard...
//
//      Puts up a dialog at the end of manual account creation to inform the user
how to create
// investments. The iWhichTip variable is an index into the rgTips array.
//--------------------------------------------------------------------------
-------
void DoNewAccountTip(CPortfolioControl *pOCX, int iWhichTip)
{
        HKEY    hKey;
        HRESULT hr;
        DWORD   dw;

        hr = RegCreateKeyEx(REGHIVE_PM, GetRscString(IDS_PREF_PREFERENCES_REGISTRY),
                        0, NULL, 0, KEY_ALL_ACCESS, NULL, &hKey, &dw);

        if ((hr != ERROR_SUCCESS) || (PrefGetLogical(hKey,
rgTips[iWhichTip].residPrefTag, FALSE) == FALSE))
        {
                if (DialogBoxParam(_Module.GetModuleInstance(),
MAKEINTRESOURCE(IDD_TIP),
                                pOCX->GetInnerWindow(), (DLGPROC)TipDialogProc,
(LPARAM)iWhichTip))
                {
                        PrefPutLogical(hKey, rgTips[iWhichTip].residPrefTag, TRUE);
                }
        }

        RegCloseKey(hKey);
}

//--------------------------------------------------------------------------
-------
// The new account wizard.
//
//      Handles new accounts linked to Money or OFX via a New Account Wizard
//--------------------------------------------------------------------------
-------
BOOL NewAccountWizard(WORD wNAWEntry, WORD wID, COFX *pOFX, void *pControl, HWND
hwndParent)
{
        BOOL                            fResult = FALSE;
        BOOL                            fNeedAddPositionTip = FALSE;
        int                                     iRet;
        CPortfolioControl       *pOCX = (CPortfolioControl *)pControl;
        CAccount                        *pNewAccount = NULL;
        SETUP_STRUCT            setup;
        INVPROPSHEETPAGE                psp[SETUP_WIZARD_PROPSHEETS];
        PROPSHEETHEADER         psh;

        if (pOCX->GetDataFileReadOnly() == TRUE)
                return (FALSE);

        if (pOCX->IsClosing() == TRUE)
                return (FALSE);

        assert(hwndParent);

        my_memset(&setup, 0, sizeof(SETUP_STRUCT));
```

```
my_memset(&psh, 0, sizeof(PROPSHEETHEADER));

setup.swatSelect = SWAT_NONE;
setup.pOCX = pOCX;
setup.wNAWEntry = wNAWEntry;
setup.wNAWModifier = NAW_MODIFIER_NONE;
setup.wID = wID;

// Check if we are doing Setup wizard or New Account wizard
switch (wNAWEntry)
{
case NAW_FIRST_TIME:
        psh.nStartPage = SWP_ACCOUNT_FTUE;
        setup.iFileVer = pOCX->GetData()->GetFileVersion();
        assert(FileHasEncryptedOFXData(pControl) == FALSE);
        break;

case NAW_NEW_ACCOUNT:
        psh.nStartPage = SWP_ACCOUNT_NEW;
        break;

case NAW_IMPORT_ACCOUNT:
        psh.nStartPage = SWP_ACCOUNT_EXISTING_IMPORT;
        break;

case NAW_REIMPORT_ACCOUNT:
case NAW_UPDATE_ACCOUNT:
        setup.swatSelect = SWAT_OFX;

        assert(pOFX);

        if ((pOFX == NULL) || (!pOFX->InitOFX(FALSE)))
                return(FALSE);

        setup.pOFX = pOFX;
        pOFX->AddRef();

        {
                CAccount        *pAccount = pOFX->GetCurAccount();

                assert(pAccount);

                // Reset update date to cause a full history update
                if (pAccount->GetBrokerID() != NULL_BROKER)
                {
                        pOFX->SetCurSessionToAccount(pAccount);

                        setup.wNAWModifier = NAW_MODIFIER_EXISTING_ACCOUNT;

                        if (wNAWEntry == NAW_UPDATE_ACCOUNT)
                                psh.nStartPage = SWP_ACCOUNT_OFX_GROUPING_1;
                        else
                                psh.nStartPage = SWP_ACCOUNT_OFX_GROUPING_3;
                }
                else
                {
                        setup.wNAWModifier = NAW_MODIFIER_NEW_ACCOUNT;
                        if (FileHasEncryptedOFXData(pControl)
                                || (setup.pOCX->GetSecurityLevel() ==
SECURITY_HIGH))
                        {
                                psh.nStartPage = SWP_ACCOUNT_OFX_1;
                        }
```

```
                                        setupwiz
                                else
                                {
                                        psh.nStartPage =
SWP_ACCOUNT_MASTER_PASSWORD;
                                }
                        }
                }

                break;

        default:
                break;
        }

        assert(hwndParent);
        psh.dwSize = PROPSHEETHEADER_V1_SIZE;
        psh.dwFlags = PSH_PROPSHEETPAGE | PSH_WIZARD | PSH_NOAPPLYNOW;
        psh.hwndParent = hwndParent;
        psh.hInstance = OCX(pOCX)->GetInstance();
        psh.pszIcon = NULL;
        psh.nPages = sizeof(psp)/sizeof(INVPROPSHEETPAGE);

        InitAccountWizardPSP(pOCX, &setup, psp);
        psh.ppsp = (PROPSHEETPAGE*)(&psp);

        OCX(pOCX)->ModalDialog(TRUE);

        // Run the Setup Wizard PropSheets
        iRet = PropertySheet(&psh);
        if (iRet == -1) // PropertySheet may actually fail
        {
                DWORD   dwError;
                TCHAR   szErrorMessage[256];
                TCHAR   szDisplayMessage[256*2];
                dwError = GetLastError();
                FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, NULL, dwError, 0,
szErrorMessage, 256, NULL);
                my_strcpyn(szDisplayMessage, "There has been a system error while
creating a New Account. ", 256*2);
                my_strcatn(szDisplayMessage, szErrorMessage, 256*2);
                MessageBox(NULL, szDisplayMessage, "Error", MB_OK |
MB_ICONINFORMATION);
        }

        OCX(pOCX)->ModalDialog(FALSE);

        if (setup.pOFXDownload)
                ReleaseInterface(setup.pOFXDownload);

        if (iRet == TRUE)
        {
                switch (setup.swatSelect)
                {
                case SWAT_REGULAR:
                case SWAT_WATCH:
                        setup.pAds->ulCodes |= (DLG_OK | DLG_REFRESH);

                        // AddAccount will prompt if there are too many portfolios
                        setup.pAds->pData =
GET_DATA(pOCX)->AddAccount(GET_DATA(pOCX)->GetAccountCount(),

setup.pAds->szAccountName,
```

```
setup.pAds->fWatch,
setup.pAds->aoOrigin,
setup.pAds->fCash);

                        pNewAccount = setup.pAds->pData ?
setup.pAds->pData->CastToAccount() : 0;

                        if (pNewAccount)
                        {
                                GET_ROAMING(pOCX)->StartBatch();

                                GET_ROAMING(pOCX)->AddAccountToBatch(pNewAccount);

                                switch (setup.swatSelect)
                                {
                                case SWAT_REGULAR:
                                        // Create the cash account if needed
                                        if (setup.pAds->fCash)
pNewAccount->CreateCashPosition(setup.pAds->dblCash);

                                        if (!setup.pAds->fWatch)
                                                setup.fLaunchBuyDialog = TRUE;

                                        fNeedAddPositionTip = TRUE;
                                        break;

                                case SWAT_WATCH:
                                        SetupWatchAccount(&setup, pNewAccount);
                                        break;
                                }


GET_ROAMING(pOCX)->UpdateAccountInBatch(pNewAccount);

                                GET_ROAMING(pOCX)->EndBatch();

                                fResult = TRUE; // Created an account
                        }
                        break;

                case SWAT_YAHOO:
                case SWAT_QUICKEN_COM:
                        setup.pWeb->SetHWndParent(hwndParent);
                        fResult = setup.pWeb->ImportData(TRUE);

                        break;

                case SWAT_QUICKEN:
                        if (setup.pQuicken)
                        {
                                setup.pQuicken->SetHWndParent(hwndParent);
                                fResult = setup.pQuicken->ImportData(TRUE);
                        }
                        break;

                case SWAT_MONEY_IMPORT:
                case SWAT_MONEY_LINK:
                        if (setup.pMoney)
                        {
                                setup.pMoney->SetHWndParent(hwndParent);
```

```
                                        setupwiz
                                fResult = setup.pMoney->ImportData(TRUE);
                        }
                        break;

                case SWAT_OFX:
                        if (setup.pOFX)
                        {
                                setup.pOFX->SetHWndParent(hwndParent);
                                fResult = setup.pOFX->ImportData(TRUE);
                        }
                        break;

                default:
                        break;
                }

                // We need to get historical prices on the securities we imported.
                GET_DATA(pOCX)->SetHaveHistoricalPrices(FALSE);
        }
        else if (wNAWEntry == NAW_FIRST_TIME)
        {
                // user didn't create an account. Display a tip to tell him how to
later.
                DoNewAccountTip(OCX(pControl), TIP_ACCOUNT);
        }

        //Destroy all unused objects
        DeleteInterface(setup.pAds);
        ReleaseInterface(setup.pQuicken);
        ReleaseInterface(setup.pMoney);
        ReleaseInterface(setup.pOFX);
        ReleaseInterface(setup.pWeb);

        // All pointers should be freed here.
        assert(!setup.pOFX && !setup.pMoney && !setup.pQuicken && !setup.pAds &&
!setup.pWeb);

        if (setup.fLaunchBuyDialog && pNewAccount)
        {
                CPortList *pList = GET_LIST(pOCX);

                // Turn on display of watch accounts if they just created one and
want to add something to it
                if (!pList->GetShowWatch() && pNewAccount->IsWatch())
                        pList->SetShowWatch(TRUE);

                if ((pList->GetFilter()) >= FIRST_ACCOUNT_VIEW)
                        pList->SetFilter(pNewAccount->GetAccountNumber());

                // Update to get the account added to the listview and give the user
feedback that
                // something happened.
                pList->Refresh(TRUE);
                int iRow = pList->FindDataRecord(pNewAccount);
                if (iRow != -1)
                        pList->SelectItem(iRow, TRUE);

                pOCX->BuyDialog(TT_BUY);

                // add mDisplay a tip to tell him more investments later.
                DoNewAccountTip(OCX(pControl), TIP_INVESTMENT);
        }
```

```
        return (iRet == TRUE);
}
```

EXHIBIT B

Copy of "setupwiz.h" Computer Code

```
#ifndef SETUPWIZ_INCLUDED
#define SETUPWIZ_INCLUDED

#include "acctdlg.h"
#include "import.h"
#include "quicken.h"
#include "money.h"
#include "web.h"
#include "ofx.h"
#include "download.h"

#define TIP_ID_PROP                                    "TIPID"
#define SETUPPROP_PROP_NAME                          "SETUPPROP"
#define TIP_PROP_NAME                                "TIP"

typedef struct _NAW_TIP_STUCT
{
        RESID   residPrompt;
        RESID   residPrefTag;
        RESID   residBitmap;

} NAW_TIP_STUCT;

enum TIPS {
        TIP_ACCOUNT = 0,
        TIP_INVESTMENT,
        TIP_ROAMING
        };


enum SWP_ID {
        SWP_FIRST = 0,
        SWP_ACCOUNT_FTUE = SWP_FIRST,
        SWP_ACCOUNT_NEW,
        SWP_ACCOUNT_EXISTING_IMPORT,
        SWP_ACCOUNT_REGULAR,
        SWP_ACCOUNT_WATCH,
        SWP_ACCOUNT_WEB,
        SWP_ACCOUNT_WEB_LIST,
        SWP_ACCOUNT_WEB_FINISH,
        SWP_ACCOUNT_MONEY_1,
        SWP_ACCOUNT_MONEY_IMPORT,
        SWP_ACCOUNT_MONEY_FINISH,
        SWP_ACCOUNT_QUICKEN,
        SWP_ACCOUNT_QUICKEN_FINISH,
        SWP_ACCOUNT_MASTER_PASSWORD,
        SWP_ACCOUNT_OFX_1,
        SWP_ACCOUNT_OFX_NOGROUPING,
        SWP_ACCOUNT_OFX_GROUPING_1,
        SWP_ACCOUNT_OFX_GROUPING_2,
        SWP_ACCOUNT_OFX_GROUPING_3,
        SWP_ACCOUNT_OFX_FINISH,
        SWP_LAST = SWP_ACCOUNT_OFX_FINISH,
        SETUP_WIZARD_PROPSHEETS
        };

enum SW_ACCT_TYPE {
        SWAT_NONE = 0,
        SWAT_REGULAR,
        SWAT_WATCH,
        SWAT_IMPORT,
        SWAT_OFX,
        SWAT_MONEY,
```

```
        SWAT_MONEY_IMPORT,
        SWAT_MONEY_LINK,
        SWAT_QUICKEN,
        SWAT_YAHOO,
        SWAT_QUICKEN_COM
        };

// New account wizard entry points
enum NAWEntry {
        NAW_FIRST_TIME = 0,
        NAW_NEW_ACCOUNT,
        NAW_IMPORT_ACCOUNT,
        NAW_UPDATE_ACCOUNT,
        NAW_REIMPORT_ACCOUNT,
        };

// New account wizard entry point modifiers
#define NAW_MODIFIER_NONE                               0x0000
#define NAW_MODIFIER_NEW_ACCOUNT                0x0100
#define NAW_MODIFIER_EXISTING_ACCOUNT   0x0200

// Structure for getting account setup information
typedef struct _SETUP_STRUCT
{
        // Flow control variables
        WORD                            wNAWEntry;                      // Where we
started the wizard
        WORD                            wNAWModifier;           // Things that
affect the place we start
        WORD                            wID;                            //
Initiating command

        SW_ACCT_TYPE            swatSelect;                     // what kind of
account to add

        // Setup Wizard data
        int                                     iFileVer;                       //
Version of data loaded

        // Regular Acct data
        ACCTDLG_STRUCT          *pAds;
        BOOL                            fLaunchBuyDialog;       // launch "Record a
Buy" dialog when we're done?

        // Money Acct data
        BOOL                            fClosing;                       // TRUE if
we are editing and return should not mean next
        BOOL                            fNext;                          // TRUE if
NEXT button is enabled.
        CMoney                          *pMoney;

        // Quicken Acct data
        CQuicken                        *pQuicken;                      // Quicken
import object.

        // Web Acct data
        CWeb                            *pWeb;                          // Web
import object

        // OFX Acct data
        COFX                            *pOFX;                          // OFX
import object.
        CDownload                       *pOFXDownload;          // Current broker
```

```
profile download.

        // Master Password
        CPortfolioControl       *pOCX;
        CHAR                            szMasterPass[MAX_STRING];

} SETUP_STRUCT;

#define nDlgBrandLeft           350
#define nDlgBrandTop            17
#define nDlgBrandWidth          84
#define nDlgBrandHeight         34


BOOL AddWatchPosition(CAccount *pAccount, TCHAR *pszTicker, CDate *pDate, double
dblUnits, double dblPrice);

void DoNewAccountTip(CPortfolioControl *pOCX, int iWhichTip);

BOOL NewAccountWizard(WORD wNAWEntry, WORD wID, COFX *pAccount, void *pControl, HWND
hwndParent);


#endif
```